Katherine Vidal (SBN 194971 / vidal@fr.com)
Betty H. Chen (SBN 290588 / bchen@fr.com)
Matthew R. McCullough (SBN 301330 / mccullough@fr.com)
FISH & RICHARDSON P.C.
500 Arguello Street, Suite 500
Redwood City, CA 94063
Telephone: (650) 839-5070
Facsimile: (650) 839-5071

OF COUNSEL:
Thomas M. Melsheimer (melsheimer@fr.com)
Michael A. Bittner (bittner@fr.com)
1717 Main Street, Suite 5000
Dallas, TX 75201
Telephone: (214) 747-5070
Facsimile: (214) 747-2091

John Brinkmann (brinkmann@fr.com)
1221 McKinney Street, Suite 2800
Houston, TX 77010
Telephone: (713) 654-5300
Facsimile: (713) 652-0109

Attorneys for Defendant
APPLE INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

(SAN FRANCISCO DIVISION)

REALTIME DATA, LLC D/B/A/ IXO, Plaintiff, v. APPLE INC., Defendant(s).	Case No. 4:16-cv-02595-JD **DEFENDANT APPLE INC.'S INVALIDITY CONTENTIONS**

Pursuant to Rule 3-3 and 3-4 of the Local Patent Rules of the Northern District of California and the Court's Scheduling Order (Dkt. No. 61), Defendant Apple Inc. ("Defendant" or "Apple") provides Plaintiff Realtime Data LLC d/b/a IXO ("Plaintiff" or "Realtime") with notice of its

Invalidity Contentions with respect to those claims asserted against it, collectively being claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of U.S. Patent No. 7,181,608 ("the '608 patent"), claims 1-6, 8-9, 11-13, and 15-16 of U.S. Patent No. 8,090,936 ("the '936 patent"), and claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-109 of U.S. Patent No. 8,880,862 ("the '862 Patent") (collectively and respectively, "the asserted claims" and "the Patents-in-Suit") asserted by Plaintiff in its Disclosure of Asserted Claims and Infringement Contentions under Patent Rule 3-1.

I. RESERVATIONS

A. General Reservations

Apple relies on and incorporate by reference, as if originally set forth herein, all invalidity positions, and all associated prior art and claim charts, asserted against Realtime in any reexamination proceeding, or by any present or former defendants in any of Realtime's lawsuits, or by potential or actual licensees to the Patents-in-Suit. Specifically, Apple relies on and incorporates by reference, as if originally set forth herein, all invalidity positions asserted against Realtime in reexamination proceedings IPR2016-01365, IPR2016-01366, IPR2016-01737, IPR2016-01738, and IPR2016-01739. Moreover, Apple reserves the right, to the extent permitted by the Court and the applicable statutes and rules, to supplement these Contentions based on prior art currently known to Realtime, including prior art identified or provided to Realtime by any present or former defendant or any third parties.

Consistent with Patent Rule 3-6, Apple reserves the right to amend these Invalidity Contentions. The information and documents that Apple produces are provisional and subject to further revision. Apple reserves the right to amend or supplement these disclosures and the subsequent document production should Realtime: 1) provide any information that it failed to provide in its Patent Rule 3-1 and 3-2 disclosures; 2) amend its Patent Rule 3-1 or 3-2 disclosures in any way; or 3) attempt to rely upon any information at trial, in a hearing, or during a deposition that it failed to provide in its Patent Rule 3-1 and 3-2 disclosures.

Apple provides the information below, as well as the accompanying production of documents, for the sole purpose of complying with Patent Rules 3-3 and 3-4. The information provided shall not be deemed an admission regarding the scope of any claims or the proper construction of those claims or any terms contained therein. Nothing contained in these Invalidity Contentions should be understood or deemed to be an express or implied admission or contention with respect to the proper construction of any terms in the asserted claim, or with respect to the alleged infringement of that claim.

B. Ongoing Discovery

Furthermore, because only limited discovery has occurred and because Apple continues its search for and analysis of relevant prior art, Apple reserves the right to revise, amend, and/or supplement the information provided herein, including identifying, charting, and relying on additional references, should Apple's further search and analysis yield additional information or references, consistent with the Patent Local Rules and the Federal Rules of Civil Procedure.

Apple's Invalidity Contentions are based upon information reasonably available to Apple as of the date of these contentions. Because discovery is ongoing Apple expressly reserves the right to clarify, alter, amend, modify, or supplement these Invalidity Contentions, to identify additional prior art, and to rely on additional information, tangible things, and testimony obtained during discovery, including discovery obtained from third parties.

Discovery is in its infancy and is ongoing, and Apple's prior art investigation and third-party discovery is therefore not yet complete. Apple reserves the right to present additional items of prior art under 35 U.S.C. §§ 102(a), (b), (e), and/or (g), and/or 103 located during the course of discovery or further investigation. For example, Apple may issue subpoenas to third parties believed to have knowledge, documentation, and/or corroborating evidence concerning some of the prior art listed herein and/or additional prior art. These third parties include without limitation the authors, inventors, or assignees of the references listed in these disclosures. In addition, Apple

reserves the right to assert invalidity under 35 U.S.C. § 102(c), (d), or (f) to the extent that discovery or further investigation yield information forming the basis for such invalidity.

Similarly, Apple has not had the opportunity to take any depositions of the patent applicants named on the face the Patents-in-Suit or other persons having relevant information. Apple reserves the right to revise, amend or supplement these contentions pursuant to Federal Rule of Civil Procedure 26(e) and the Orders of record in this matter to the extent appropriate in light of further investigation and discovery regarding the defenses, the review and analysis of expert witnesses, or supplemental contentions by Realtime.

C. Claim Construction

Apple reserves the right to revise their ultimate contentions concerning the invalidity of the asserted claims of the Patents-in-Suit, which may change depending upon any findings as to the priority date of those claims and/or positions that Realtime or expert witness(es) may take concerning infringement and/or invalidity issues. Apple does not waive the right to contest any claim constructions or to take positions during claim construction proceedings that have yet to occur that may be inconsistent with the invalidity contentions herein. Consequently, Apple also reserves the right to amend or supplement these Invalidity Contentions in the event that the claims are construed differently at some point in the future.

Apple does not necessarily adopt Realtime's positions on the scope or construction of the claims. In certain instances, Apple has applied the claims to the prior art in view of Realtime's allegations, admissions, or positions for purposes of these contentions only. As such, Apple's Invalidity Contentions are not adoptions or admissions by Apple as to the accuracy of Realtime's allegations, admissions, or positions. Accordingly, these contentions are made in the alternative, are not necessarily intended to be consistent with each other, and should not be otherwise construed.

Apple expressly reserves the right to take positions with respect to future claim construction or infringement issues that are inconsistent with, or even contradictory to, the claim

construction or infringement positions expressed or implied in the Invalidity Contentions set forth herein.

D. Realtime's Infringement Contentions

Realtime's disclosures under Patent Rules 3-1 and 3-2 are deficient in numerous respects. For example, Realtime has failed to specifically identify where each element of each claim is found within each Accused Instrumentality as required by Patent Rule 3-1. Because such deficiencies may lead to further grounds for invalidity, Apple specifically reserves the right to modify, amend, or supplement their contentions as Realtime modifies, amends, or supplements its disclosures under Patent Rules 3-1, 3-2, and/or 3-6, and/or produces documents in discovery.

Additionally, Realtime has presented no substantive contentions for indirect infringement, i.e., active inducement or contributory infringement. Realtime has not, for example, provided detailed contentions that identify how Apple allegedly induces direct infringement of the Patents-in-Suit by a third party, or how Apple allegedly contributes to the infringement of the Patents-in-Suit by a third party. Nor has Realtime provided detailed contentions regarding any alleged infringement by multiple parties pursuant to 35 U.S.C. § 271(a) (i.e., joint infringement). Nor has Realtime provided detailed contentions of any alleged infringement under the doctrine of equivalents. If Realtime is permitted to provide this and other information relating to alleged indirect infringement, joint infringement, or infringement under the doctrine of equivalents, Apple will amend and supplement these Invalidity Contentions as appropriate.

E. The Intrinsic Record

Apple further reserves the right to rely upon applicable industry standards and prior art cited in the file histories of the Patents-in-Suit and any related U.S. and foreign patent applications as invalidating references or to show the state of the art. Apple further reserves the right to rely on the patent applicants' admissions concerning the scope of the prior art relevant to the asserted patents found in, *inter alia*: the patent prosecution history for the asserted patent and any related patents and/or patent applications or reexaminations; any deposition testimony of the named patent

applicants on the asserted patent; and the papers filed and any evidence submitted by Realtime in connection with this litigation.

F. Rebuttal Evidence

Prior art not included in these Invalidity Contentions, whether known or not known to Apple, may become relevant. In particular, Apple is currently unaware of the extent, if any, to which Realtime will contend that limitations of the asserted claims of the Patents-in-Suit are not disclosed in the prior art identified herein. To the extent that such an issue arises, Apple reserves the right to identify other references that would render obvious the allegedly missing limitation(s) or the disclosed device or method.

G. Contextual Evidence

Apple's claim charts cite particular teachings and disclosures of the prior art as applied to the limitation of each of the asserted claims. However, persons having ordinary skill in the art generally may view an item of prior art in the context of his or her experience and training, other publications, literature, products, and understanding. As such, the cited portions are only examples, and Apple reserves the right to rely on uncited portions of the prior art references and on other publications and expert testimony as aids in understanding and interpreting the cited portions, as providing context thereto, and as additional evidence that the prior art discloses a claim limitation or the claimed subject matter as a whole. Apple further reserves the right to rely on uncited portions of the prior art references, other publications, and testimony, including expert testimony, to establish bases for combinations of certain cited references that render the asserted claims obvious. The references discussed in the claim charts may disclose the elements of the asserted claims explicitly and/or inherently, and/or they may be relied upon to show the state of the art in the relevant time frame. The suggested obviousness combinations are provided in the alternative to anticipation contentions and are not to be construed to suggest that any reference included in the combinations is not by itself anticipatory.

H. Invalidity Under Section 102(f) Prior Art

Apple reserves the right to assert that the asserted claims of the Patents-in-Suit are invalid under 35 U.S.C. § 102(f) in the event Apple obtains evidence that James J. Fallon, John Buck, Paul F. Pickel, and/or Stephen McEerlain, the inventors named on the asserted or related patents, did not themselves "invent" the subject matter claimed. Should Apple obtain such evidence, it will provide the name of the person(s) from whom and the circumstances under which the claimed subject matter or any part of it was derived.

I. Priority And Effective Filing Date

Apple contends that, for each Patent-in-Suit, Realtime will be unable to demonstrate that the asserted claims are entitled to claim a priority date or effective filing date earlier than the actual filing date of the application that issued as that patent. No ancestor application, including provisional application No. 60/180,114 filed on February 3, 2000, provides a disclosure sufficient under 35 U.S.C. § 112(1/a) to support such claim as required by section 119(e) or 120. *See* Section VIII below.

J. No Patentable Weight

Apple reserves the right to argue that various portions of the asserted claims, such as an intended use or result, non-functional descriptive material, and certain preamble language, are entitled to no patentable weight. Mapping of a portion of an asserted claim to a prior art reference does not represent that such portion of the claim is entitled to patentable weight when comparing the claimed subject matter to the prior art.

II. IDENTIFICATION OF PRIOR ART

At least the prior art listed below, individually or in combination, invalidates the asserted claims. *See* Patent Rule 3-3(a). Appendices A1-C38 provide detailed claim charts showing where each claim element may be found in the particular reference being charted.

Apple identifies the following items of prior art that anticipate or render obvious the asserted claims. The identification of prior art below is not exclusive, and Apple's production

pursuant to Patent Rule 3-3 contains additional references that render the asserted claims invalid. Apple reserves the right to rely upon both the listed and unlisted references produced pursuant to Patent Rule 3-3, as well as other art that may become known and/or relevant during the course of this or related litigation.

For those primary references for which detailed claim charts are provided in Appendices A1-C38, a reference to the particular Appendix Number is provided in Section IV below. References for which Appendix Numbers do not appear are additional prior art references that are either included as secondary references in charts contained in Appendices A1- C38, or are otherwise pertinent to the invalidity of the Patents-in-Suit, either alone or in combination with other references. At this time, Apple is not providing claim charts for each of these additional references, either because they are cited in conjunction with primary references for which charts have already been provided and are cited therein, and/or because these references have similar disclosure to the prior art references for which invalidity charts have been provided and/or may be used to show the state of the art.

Apple also incorporates as if fully set forth herein the complete file histories for the '608 patent, the '936 patent, and the '862 patent, including any prior art or supporting documents cited therein.

Apple not only relies upon the prior art disclosed herein, but also relies on any commercial embodiments and accompanying literature of the various assignees that correspond to the respective disclosures found within the prior art disclosed herein. The assignees' various and respective commercial embodiments and/or corresponding literature anticipate and/or render obvious the claims of the Patents-in-Suit for at least the reasons disclosed in these Invalidity Contentions and claim charts, as well as for other independent reasons found within the commercial embodiments and corresponding literature. Apple also reserves the right to rely on related patents, published applications, foreign patents or publications, and other patent documents as necessary to establish prior art status or clarify the disclosures cited.

Apple reserves the right to revise their claim charts to rely on any of these references to prove the invalidity of the claims of the Patents-in-Suit in a manner consistent with the Federal Rules of Civil Procedure, the Court's Local Rules, the Local Patent Rules and this Court's Orders.

A. Prior Art Patents And Published Applications

- U.S. Patent No. 3,490,690 to Apple
- U.S. Patent No. 4,476,526 to Dodd
- U.S. Patent No. 4,593,324 to Ohkubo
- U.S. Patent No. 4,956,808 to Aakre
- U.S. Patent No. 5,003,307 to Whiting
- U.S. Patent No. 5,101,490 to Getson
- U.S. Patent No. 5,131,089 to Cole
- U.S. Patent No. 5,142,680 to Ottman
- U.S. Patent No. 5,150,430 to Chu
- U.S. Patent No. 5,269,022 to Shinjo ("Shinjo")
- U.S. Patent No. 5,307,497 to Feigenbaum ("Feigenbaum")
- U.S. Patent No. 5,410,671 to Elgamal
- U.S. Patent No. 5,420,998 to Horning ("Horning")
- U.S. Patent No. 5,421,031 to De Bey
- U.S. Patent No. 5,432,927 to Grote
- U.S. Patent No. 5,467,087 to Chu
- U.S. Patent No. 5,481,701 to Chambers
- U.S. Patent No. 5,519,843 to Moran
- U.S. Patent No. 5,530,847 to Schieve
- U.S. Patent No. 5,557,777 to Culbert
- U.S. Patent No. 5,581,785 to Nakamura
- U.S. Patent No. 5,600,766 to Deckys

- U.S. Patent No. 5,619,698 to Lillich ("Lillich '698")
- U.S. Patent No. 5,632,024 to Yajima
- U.S. Patent No. 5,652,886 to Tupule
- U.S. Patent No. 5,671,413 to Shipman ("Shipman")
- U.S. Patent No. 5,696,897 to Dong
- U.S. Patent No. 5,699,539 to Garber
- U.S. Patent No. 5,729,228 to Franaszek
- U.S. Patent No. 5,764,994 to Craft
- U.S. Patent No. 5,790,856 to Lillich ("Lillich '856")[1]
- U.S. Patent No. 5,793,943 to Noll ("Noll")
- U.S. Patent No. 5,794,052 to Harding
- U.S. Patent No. 5,805,086 to Brown
- U.S. Patent No. 5,805,882 to Cooper
- U.S. Patent No. 5,809,295 to Straub
- U.S. Patent No. 5,812,817 to Hovis ("Hovis")
- U.S. Patent No. 5,815,705 to Slivka
- U.S. Patent No. 5,819,115 to Hoese
- U.S. Patent No. 5,828,877 to Pearce ("Pearce")
- U.S. Patent No. 5,836,013 to Greene ("Greene")
- U.S. Patent No. 5,860,083 to Sukegawa ("Sukegawa")
- U.S. Patent No. 5,884,074 to Maeda
- U.S. Patent No. 5,901,310 to Rahman ("Rahman")
- U.S. Patent No. 5,907,703 to Kronenberg
- U.S. Patent No. 5,920,896 to Grimsrud
- U.S. Patent No. 5,925,129 to Combs

[1] Lillich '698 and Lillich '856 are collectively referred to herein as "Lillich."

- U.S. Patent No. 5,930,358 to Rao
- U.S. Patent No. 5,933,630 to Ballard ("Ballard")
- U.S. Patent No. 5,940,871 to Goyal
- U.S. Patent No. 5,948,104 to Gluck
- U.S. Patent No. 5,991,542 to Han
- U.S. Patent No. 6,014,694 to Aharoni
- U.S. Patent No. 6,073,232 to Krocker ("Krocker")
- U.S. Patent No. 6,098,158 to Lay
- U.S. Patent No. 6,108,014 to Dye
- U.S. Patent No. 6,128,094 to Smith
- U.S. Patent No. 6,169,844 to Arai
- U.S. Patent No. 6,175,896 to Bui
- U.S. Patent No. 6,182,122 to Berstis
- U.S. Patent No. 6,212,632 to Surine ("Surine")
- U.S. Patent No. 6,216,225 to Yoo
- U.S. Patent No. 6,237,080 to Makinen ("Makinen")
- U.S. Patent No. 6,253,264 to Sebastian
- U.S. Patent No. 6,263,431 to Lovelace
- U.S. Patent No. 6,266,736 to Atkinson
- U.S. Patent No. 6,272,628 to Aguilar
- U.S. Patent No. 6,272,629 to Stewart
- U.S. Patent No. 6,279,092 to Franaszek
- U.S. Patent No. 6,317,818 to Zwiegincew ("Zwiegincew")
- U.S. Patent No. 6,336,161 to Watts
- U.S. Patent No. 6,370,614 to Teoman ("Teoman")
- U.S. Patent No. 6,370,631 to Dye

- U.S. Patent No. 6,374,353 to Settsu ("Settsu")
- U.S. Patent No. 6,393,584 to McLaren
- U.S. Patent No. 6,401,202 to Abgrall
- U.S. Patent No. 6,421,776 to Hillis ("Hillis")
- U.S. Patent No. 6,442,623 to Kim
- U.S. Patent No. 6,434,695 to Esfahani ("Esfahani '695")
- U.S. Patent No. 6,452,602 to Morein
- U.S. Patent No. 6,457,175 to Lerche
- U.S. Patent No. 6,473,856 to Goodwin
- U.S. Patent No. 6,564,318 to Gharda
- U.S. Patent No. 6,567,911 to Mahmoud
- U.S. Patent No. 6,622,244 to Eidson
- U.S. Patent No. 6,601,167 to Gibson
- U.S. Patent No. 6,636,963 to Stein
- U.S. Patent No. 6,732,265 to Esfahani ("Esfahani '265")[2]
- U.S. Patent No. 6,823,435 to Wisor
- U.S. Patent No. 7,190,284 to Dye
- U.S. Patent No. 8,176,288 to Dye
- U.S. Patent Application No. 2001/0039612 to Lee ("Lee")
- U.S. Patent Application No. 2004/0068646 to Stein
- PCT Application No. WO 92/17844 to Miller
- PCT Application No. WO 94/19768 to Kikinis ("Kikinis")
- PCT Application No. WO 96/13772 to Shipman
- PCT Application No. WO 97/37847 to Brown
- European Patent No. 0713176 to Voce

[2] Esfahani '695 and Esfahani '265 are collectively referred to herein as "Esfahani."

- European Patent No. 0788115 to Lee
- European Patent No. 0868063 to Berstis
- German Patent No. DE19721786 to Michael Vers ("Vers")
- G.B. Patent No. 2276257 to Ingvar ("Ingvar")
- Japanese Patent No. 06-230974 to Takashi
- Japanese Patent No. 11-316683 to Suziki

B. <u>Prior Art Non-Patent References</u>

- Abali et al., "Operating System Support for Fast Hardware Compression of Main Memory Contents," Memory Wall Workshop, June 2000
- Anyimi, "Implementing a Plug and Play BIOS Using Intel's Boot Block Flash Memory," Feb. 1995 ("Anyimi")
- Baker et al., "Lossless Data Compression for Short Duration 3D Frames in Positron Emission Tomography," Nuclear Science Symposium and Medical Imaging Conference, 1993
- M. Beck, et. al, "Linux Kernel Internals" Addison Wesley Longman (1996) ("Beck")
- D. Bennett, "Booting Linux from EPROM," Linux Journal, January 1997 ("Bennett")
- Michael Burrows et al., "On-line data compression in a log-structured file System" ("Burrows")
- Cheng et al., "Fast and highly reliable IBMLZ1 compression chip and algorithm for storage," Hot Chips V11, August 1995 ("Cheng")
- Craft, "A Fast hardware data compression algorithm and some algorithmic extension," IBM J. Res. Develop., Vol 43, Nov. 1998 ("Craft")
- Crowley et al., "Dynamic Compression During System Save Operation," IBM Technical Disclosure Bulletin, May 1, 1984

- Douglis, “One the Role of Compression in Distributed Systems” (“Douglis 1”)
- Douglis, “The Compression Cache: Using On-Line Compression to Extend Physical Memory,” Winter 1993 USENIX Conference, Jan 1993 (“Douglis 2”)[3]
- Fiala, et al., "Data Compression with Finite Windows."
- Grove “System Administration,” LINUX, Mar 1998 (“Grove”)
- Jones, “The Microsoft Interactive TV system: An Experience Report,” July 1997 (“Jones”)
- Kawaguchi et al., “A Flash-Memory Based File System,” Proceedings of the USENIX 1995 Technical Conference Proceedings, 1995
- “Magstar and IBM 3590 High Performance Tape Subsystem Technical Guide,” November 1996 (“Magstar”)
- Mealey, B, IBM, “An IP.com Prior Art Database Technical Disclosure,” January, 1992 (“Mealey”)
- Menon, “A performance comparison of RAID-5 and log-structured arrays,” IBM Almaden Research Center (“Menon”)
- Z. Palmer “Fido: A Cache That Learns to Fetch,” Proceedings of the 17th International Conference on Very Large Data Bases (Sep. 1991)
- Red Hat Linux 5.0, The Official Red Hat Linux Installation Guide (1995) (“Linux Redhat”)[4]
- Rubini, “Booting the Kernel,” Linux Journal, Jan. 1997 (“Rubini”)
- “Seagate Enters Mid-Range Tape Market with Innovative Sidewinder® 50,” PR Newswire, May 19, 1997
- “Sidewinder,” Infoworld, July 14, 1997
- Simpson et al., “A Multiple Processor Approach to Data Compression,” Proceedings of the 1998 ACM Symposium on Applied Computing, 1998

[3] Douglis 1 and Douglis 2 are collectively referred to herein as “Douglis.”
[4] Beck and Linus Redhat are collectively referred to herein as “Linux Kernel.”

- Wang, et al., "The feasibility of using compression to increase memory system perfomance," ECE Technical Reports, 1993
- Welch et al., "A Technique for High-Performance Data Compression," IEEE Computer, 1984
- Wilson, et al., "The Case for Compressed Caching in Vitrual Memory Systems," Proceedings of the USENIX Annual Technical Conference, June 1999
- Wynn, et al., "The effect of compression on performance in a demand paging operating system," The Journal of Systems and Software (2000) ("Wynn Article")
- Wynn, "The Effect of Compression on Performance in a Demand Paging Operating System," 1997 ("Wynn Thesis")[5]
- Zobel, et al., "Adding compression to a full-text retrieval system," Software--Practice and Experience (Aug. 1995)

C. <u>Prior Art Offered For Sale And/Or Publicly Used Or Known</u>

Pursuant to Local Patent Rule 3-3(a), Apple provides the following information regarding prior art which was the subject of a commercial offer of sale and/or in public use prior to the earliest permissible priority date of the asserted patents. The following table further includes information regarding derivation, and also regarding prior art which was made in this country and not abandoned, suppressed, or concealed and/or otherwise qualifies as prior art under 35 U.S.C. § 102, including Section 102(g), with an effective date prior to the earliest asserted conception date of the asserted patents.

- Apple's 68K Operating System ("The 68K System"). On information and belief, this product package was invented and purchased, evaluated, compared, tested, implemented, made, used (publicly and/or commercially), disclosed, offered for sale, and/or sold by Apple in the United States at least as early as May 1995.

[5] The Wynn Article and the Wynn Thesis are collectively referred to herein as "Wynn."

- Apple's New World Mac Operating System ("The New World Mac System"). On information and belief, this product package was invented and purchased, evaluated, compared, tested, implemented, made, used (publicly and/or commercially), disclosed, offered for sale, and/or sold by Apple in the United States at least as early as December 1998.
- The Linux Operating System ("Linux OS"). On information and belief, this product package was invented and purchased, evaluated, compared, tested, implemented, made, used (publicly and/or commercially), disclosed, offered for sale, and/or sold by Apple in the United States at least as early as September 1991.
- The Microsoft Windows 2000 Operating System ("Windows 2000"). On information and belief, this product package was invented, evaluated, compared, tested (including beta testing), implemented, made, used (publicly and/or commercially via beta or early release), and/or disclosed, by Microsoft in the United States prior to as February 3, 2000.

D. Admitted Prior Art

The applicant for the Patents-in-Suit has expressly or implicitly admitted that certain elements recited in asserted claims of the Patents-in-Suit were known in the prior art and thus part of the state of the art. These admissions include, but are not limited to, the following prior art concepts. Apple reserves the right to identify additional examples of admitted prior art and to further support the admissions identified below by relying on additional portions of the patent specification, statements made during the prosecution history of the Patents-in-Suit and the prosecution history of any related applications, and any statements by the Plaintiff or the patent applicants.

Patent	Admissions by Patent Applicant
608	The '608 patent admits that the following was already known to persons of skill in the art before the alleged invention:

Patent	Admissions by Patent Applicant
	• the concept of "accelerated data storage," including "receiving a digital data stream at a data transmission rate which is greater than the data storage rate of the target storage device, compressing the input data stream at a compression rate that increases the effective storage rate of the target storage device and storing the compressed data in the target storage device" as discussed at 5:47-54 and 24:13-34; • the concept of "accelerated data retrieval," including "retrieving a compressed digital data stream from a target storage device at a rate equal to, e.g., the data access rate of the target storage device and then decompressing the compressed data at a rate that increases the effective data access rate of the target storage device" that mitigates "the traditional bottleneck associated with, e.g., local and network disk accesses" as discussed at 5:62-6:2; • certain "data compression/decompression techniques" as disclosed in U.S. Patent. No. 6,195,024, which "are suitable for compressing and decompressing at rate, which provide accelerated data storage and retrieval" as discussed at 6:16-20; • known Ultra DMA, SCSI, Serial Storage Architecture, and Fibre Channel disk interfaces, as discussed at 6:28-32; • the American National Standard for Information Systems (ANSI) AT Attachment Interface (ATA/ATSPI-4), as discussed at 6:32-34; • known standards such as the PCI (Peripheral Component Interconnect) bus interface for interfacing with a computer system, as discussed at 6:41-43; • known storage devices, including "all forms of random access memory, magnetics and optical tape, magnetics and optical disks, along with various other forms of solid-state mass storage devices," as discussed at 6:53-56; • lossless encoding techniques "such as run length, Huffman, Lempel-Ziv Dictionary Compression, arithmetic code, data compaction, and data null suppression," as discussed at 24:60-64 and 26:66-27:7; • the concept of an "option to enable/disable any one or more of the encoders," as discussed at 25:3-13; • the concept of a "threshold limit" that may be "specified as any value inclusive of the data expression, no data compression or expansion, or any arbitrarily desired compression limit," as discussed at 25:50-53; and • certain "methods known by those skilled in the art to extract the data compression type descriptor associated with the data block," as discussed at 26:55-60. By incorporating U.S. Patent 6,195,024 by reference, the '608 patent admits that the following was also already known to persons of skill in the art before the alleged invention as disclosed in the referenced patent: • the concept of selecting an appropriate lossless data compression technique (disclosed in U.S. Patent No. 5,467,087 to Chu), as discussed at '024, 2:63-3:41 and Fig. 1;

Patent	Admissions by Patent Applicant
	• the concept of using multiple dictionaries within a single encoding process to aid in reducing data dependency on a single encoding technique (disclosed in U.S. Patent No. 5,243,241 to Seroussi), as discussed at '024, 3:42-60; • the concept of using a plurality of code tables to increase compression rate (disclosed in U.S. Patent No. 5,171,393 to Nakano), as discussed at '024, 3:61-4:8; • techniques for dividing uncompressed data into a plurality of stream for subsequent compression by a plurality of encoders (disclosed in U.S. Patent No. 5,809,176 to Yajima), as discussed at '024, 4:9-15; • systems for parallel compression of a data stream (disclosed in U.S. Patent Nos. 5,583,500 and 5,471,206 to Allen), as discussed at '024, 4:16-22; • two-stage lossless compression processes (disclosed in U.S. Patent No. 5,627,534 to Craft), as discussed at '024, 4:23-32; • the concept of an adaptive threshold technique for achieving a constant bit rate (disclosed in U.S. Patent No. 5,799,110 to Isrealsen), discussed at '024, 4:33-40; and • a method of applying either lossy or lossless compression (such as run-length and Huffman encoding) to achieve a desire level of quality (disclosed in U.S. Patent No. 5,819,215 to Dobson), discussed at '024, 4:41-48.
936	Because the '936 patent shares a common specification with the '608 patent above, the '936 patents admits that the same prior art discussed above was already known to persons of skill in the art before the alleged invention.
862	Because the '862 patent shares a common specification with the '608 patent above, the '862 patents admits that the same prior art discussed above was already known to persons of skill in the art before the alleged invention.

E. The Applicants' Own Prior Patent And Published Applications

As noted, no asserted claim of any Patent-in-Suit is entitled to be backdated before the actual filing date of the application that issued as that patent. The patents of the applicants and/or the Realtime, published or issues prior to the filing date of any Patent-in-Suit, thus qualify as Sec. 102(b) prior art, and state of the art, to each of the asserted claims:

III. PRIOR ART CLAIM CHARTS

Pursuant to Patent Rule 3-3(c), the claim charts attached hereto as Appendices A1-C38 identify specifically where each limitation of each claim is found in each prior art reference. Apple

expressly reserves the right to supplement or amend these contentions pursuant to Patent Rule 3-6 after the Court's *Markman* ruling or if Realtime is permitted to amend or alter its infringement contentions or its position on claim construction in any way. Further, to the extent that Apple applies Realtime's constructions (and to the extent these constructions can be discerned), Apple does not concede in any way that those constructions are correct, and instead expressly reserves the right to oppose those constructions at the appropriate time specified in the Court's Scheduling Order. Nothing in these disclosures should be construed as an admission by Apple.

In its Invalidity Charts, Apple has referred to relevant, representative portions of the cited prior art. The absence of any specific or express reference to any claim term or claim element in these charts should not be construed as an admission that any corresponding limitation is lacking either expressly or inherently in the prior art reference. There may be additional support or other grounds for Apple's contentions that such prior art satisfies a particular claim element, and Apple reserves the right to supplement these invalidity contentions with such information. For example, persons of ordinary skill in the art at the time of the filing of the Patents-in-Suit knew to read prior art references as a whole, and in the context of other publications and literature and the general knowledge in the field. Apple may rely on all such information, including uncited portions of the prior art references listed herein, and on other publications and expert testimony, to provide context and as aids to understanding and interpreting the listed references, or to establish that a person of ordinary skill in the art would have been motivated to modify or combine any of the cited references so as to render the claims obvious. Additionally, citations to a particular figure in a prior art reference encompass all text relating to the figure, and citations to text encompass all figures relating to or referred to by that text.

Moreover, many of the prior art systems and apparatuses were sold prior to the earliest asserted priority dates and documentation regarding these prior art systems are not in the possession, custody, or control of Apple. Instead, these documents are currently or will be the

subject of third-party discovery. Apple expressly reserves the right to supplement their contentions to include further information obtained in discovery.

IV. PRIOR ART UNDER 35 U.S.C. § 102

The prior art listed above anticipates the asserted claims of the Patents-in-Suit either expressly or inherently as understood by a person having ordinary skill in the art. The specific anticipation assertions with respect to each claim are set forth in the accompanying claim charts, Appendices A1-C38. Appendices A1-A38 are charted against the '608 patent. Appendices B1-B38 are charted against the '936 patent. Appendices C1-C38 are charted against the '862 patent.

Appendix No. (A v. '608 / B v. 936 / C v. '862)	**Reference**
A1 / B1/ C1	Esfahani and the New World Mac System
A2 / B2 / C2	Lillich and the 68K System
A3 / B3 / C3	Ballard
A4 / B4 / C4	Feigenbaum
A5 / B5 / C5	Greene
A6 / B6 / C6	Hillis
A7 / B7 / C7	Horning
A8 / B8 / C8	Hovis
A9 / B9 / C9	Ingvar
A10 / B10 / C10	Kikinis
A11 / B11 / C11	Korcker
A12 / B12 / C12	Lee
A13 / B13 / C13	Makinen
A14 / B14 / C14	Noll
A15 / B15 / C15	Pearce
A16 / B16 / C16	Rahman

Appendix No. (A v. '608 / B v. 936 / C v. '862)	Reference
A17 / B17 / C17	Settsu
A18 / B18 / C18	Shinjo
A19 / B19 / C19	Shipman
A20 / B20 / C20	Sukegawa
A21 / B21 / C21	Surine
A22 / B22 / C21	Teoman
A23 / B23 / C21	Vers
A24 / B24 / C22	Zwiegincew
A25 / B25 / C23	Anyimi Publication
A26 / B26 / C26	Bennett Publication
A27 / B27 / C27	Burrows Publication
A28 / B28 / C28	Cheng Publication
A29 / B29 / C29	Craft Publication
A30 / B30 / C30	Douglis Publication
A31 / B31 / C31	Grove Publication
A32 / B32 / C32	Jones Publication
A33 / B33 / C33	Magstar Publication
A34 / B34 / C34	Mealey Publication
A35 / B35 / C35	Menon Publication
A36 / B36 / C36	Rubini Publication
A37 / B37 / C37	Wynn Publication
A38 / B38 / C38	Linux Kernel

Apple has endeavored to identify the most relevant portions of identified references. The references may contain additional support, however, for a particular claim element. Apple may

rely on uncited portions of the prior art references and/or other publications and fact or expert testimony to provide context and as aids to understanding and interpreting the portions that are cited.

The references discussed in the claim charts may disclose the elements of the claims explicitly and/or inherently, and/or they may be relied upon to show the state of the art in the relevant time frame. The suggested obviousness combinations are provided in the alternative to Apple's anticipation contentions and are not to be construed to suggest that any reference included in any combination is not by itself anticipatory. Also, the suggested obviousness combinations are provided as examples, and it should be understood that other combinations of the prior art disclosed and cited herein could be used in such combinations.

V. PRIOR ART UNDER 35 U.S.C. § 103

To the extent that the asserted claims are not rendered invalid purely on anticipatory grounds or are not obvious in light of the general knowledge in the field and of one skilled in the art, the prior art references render obvious the asserted claims as discussed in detail below. These combinations are not exclusive, and Apple reserves the right to supplement the obviousness arguments listed below, using any references listed above in Section II and any references that may become known to Apple during the course of discovery.

VI. MOTIVATION FOR COMBINING IDENTIFIED PRIOR ART

Pursuant to Local Patent Rule 3-3, Apple has included this section discussing motivation to combine. The Supreme Court, however, has rejected the idea that a "teaching, suggestion, or motivation to combine" is a prerequisite for proving obviousness. *See KSR Int'l Co. v. Teleflex, Inc.*, 127 S. Ct. 1727, 1739-40 (2007) (rejecting the Federal Circuit's "rigid" application of the teaching, suggestion, or motivation to combine test, and instead espousing an "expansive and flexible" approach). Indeed, the Supreme Court held that a person of ordinary skill in the art is "a person of ordinary creativity, not an automaton" and "in many cases a person of ordinary skill in

the art will be able to fit the teachings of multiple patents together like pieces of a puzzle." *Id.* at 1742.

Subject to the reservation of rights based on Apple's present understanding of the claims of the Patents-in-Suit, and the apparent constructions that Realtime is asserting based on its Infringement Contentions, the prior art references identified above in Section II, which are charted in Appendices A1-C38, each anticipate and/or render obvious the asserted claims of the Patents-in-Suit.

Should any individual prior art reference be deemed not to disclose, explicitly or inherently, any element of a claim, Apple contends in the alternative that such individual prior art references identified above in Section II (and charted in Appendices A1- C38) can be combined with the specific references identified below for individual claim elements, as indicated in many of the charts in Appendices A1- C38. Specific motivations for such combinations are indicated in the claim charts and below.

Further, to the extent that the claims are not rendered invalid purely on anticipatory grounds or are not obvious in light of the general knowledge in the field and of one skilled in the art, the following prior art combinations of references render obvious the asserted claims of the Patents-in-Suit. To the extent a cited prior art reference is deemed not to anticipate or render obvious a claim for failing to teach or suggest one or more elements of that claim, that claim would nonetheless have been obvious to one of ordinary skill in the art at the time of the invention by the combination of the cited prior art reference with one or more other prior art references disclosing the missing claim element. Listed below are examples of references disclosing each general claim element in the asserted claims of the Patents-in-Suit:

Claim Element		**References Disclosing Claim Element for Combination**
Pre	a boot device, including the following components:	• Ballard • Bennett • Esfahani • Greene

Claim Element		References Disclosing Claim Element for Combination
		• Hillis • Kikinis • Lillich • Rubini • Settsu • Shinjo • Sukegawa
1	a processor or central processing unit, including initializing a processor or central processing unit	• Ballard • Douglis • Esfahani • Greene • Hillis • Horning • Kikinis • Krocker • Lillich • Makinen • Noll • Pearce • Rubini • Settsu • Shinjo • Sekegawa
2	a controller, including a data storage controller or digital signal processor	• Esfahani • Hillis • Kikinis • Lillich • Noll • Settsu • Shipman • Sukegawa
3	memory, including volatile memory or cache memory	• Ballard • Bennett • Douglis • Esfahani • Greene • Horning • Kikinis • Krocker • Lillich • Rubini

Claim Element		References Disclosing Claim Element for Combination
		• Settsu • Shipman • Sukegawa • Surine
4	memory, including non-volatile memory or physical memory	• Ballard • Bennett • Esfahani • Greene • Horning • Kikinis • Krocker • Lillich • Makinen • Rubini • Settsu • Shinjo • Shipman • Sukegawa • Surine
5a	boot data stored in memory, including in either volatile or non-volatile memory	• Ballard • Bennett • Esfahani • Feigenbaum • Greene • Hillis • Kikinis • Lee • Lillich • Linux Kernel • Makinen • Rahman • Rubini • Settsu • Shinjo • Shipman • Sukegawa • Surine • Teoman
5b	loading boot data for an operating system, application, or portions of the same, which may include a	• Ballard • Bennett • Esfahani

Claim Element		References Disclosing Claim Element for Combination
	plurality of files	• Feigenbaum • Greene • Hillis • Krocker • Lillich • Linux Kernel • Noll • Rubini • Settsu • Shinjo • Sukegawa • Teoman
5c	wherein such boot data is compressed	• Bennett • Esfahani • Greene • Hillis • Jones • Kikinis • Lee • Lillich • Linux Kernel • Makinen • Noll • Rahman • Rubini • Vers • Zwiegincew
6	such compression including lempel-ziv compression, dictionary compression, and/or compression carried out by a plurality of encoders	• Burrows • Cheng • Craft • Douglis • Esfahani • Greene • Lillich • Linux Kernel • Kikinis • Magstar • Menon • Pearce • Rahman • Vers

Claim Element		References Disclosing Claim Element for Combination
		• Wynn • Zwiegincew
7	maintaining a list of boot data	• Ballard • Bennett • Esfahani • Greene • Hillis • Krocker • Lee • Lillich • Linux Kernel • Makinen • Rubini • Settsu • Sukegawa
8	updating a list of boot data, including additions, deletions, or disassociations related to the list	• Ballard • Esfahani • Krocker • Lillich • Linux Kernel • Makinen • Settsu • Sukegawa
9	loading boot data via direct memory access	• Ballard • Esfahani • Greene • Hillis • Kikinis • Lillich • Makinen • Rubini • Settsu • Sukegawa
10	preloading boot data, including loading into a cache and loading prior to completion of initialization of the processor or central processing unit	• Bennett • Esfahani • Feigenbaum • Greene • Hillis • Lillich

Claim Element		References Disclosing Claim Element for Combination
		• Linux Kernel • Kikinis • Settsu • Sukegawa
11	accessing and/or decompressing boot data, including to boot/load the operating system, application, or portions thereof	• Bennett • Esfahani • Feigenbaum • Greene • Hillis • Kikinia • Lee • Lillich • Linux Kernel • Makinen • Rahman • Rubini • Settsu • Shipman • Sukegawa • Surine • Vers
12	loading, accessing, and/or decompressing at an increased rate, including increasing effective rate of cache or decreasing boot/load time relative to uncompressed data	• Ballard • Bennett • Cheng • Craft • Esfahani • Feigenbaum • Hillis • Lee • Lillich • Linux Kernel • Kikinis • Settsu • Sukegawa • Surine • Vers • Wynn • Zwiegincew
13	a data compression engine that compresses and decompresses boot data	• Greene • Lillich • Linux Kernel

Claim Element		References Disclosing Claim Element for Combination
		• Kikinis • Zwiegincew

These combinations are not exclusive, and Apple reserves the right to supplement the obviousness arguments listed below, using any references listed above in Section II and any references that may become known to Apple during the course of discovery.

The United States Supreme Court recently clarified the standard for what types of inventions are patentable. *KSR Int'l*, 127 S. Ct. 1727. In particular, the Supreme Court emphasized that inventions arising from ordinary innovation, ordinary skill, or common sense should not be patentable. *Id.* at 1732, 1738, 1742-1743, 1746. In that regard, a patent claim may be obvious if the combination of elements was obvious to try or there existed at the time of the invention a known problem for which there was an obvious solution encompassed by the patent's claims. In addition, when a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, Section 103 likely bars its patentability.

"[T]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." *Id.* at 1731. Because the Patents-in-Suit simply arrange old elements with each performing the same function it had been known to perform and yields no more than what one would expect from such an arrangement, the combination is obvious. *Id.* at 1742. The asserted claims are therefore invalid under 35 U.S.C. § 103 because they do nothing more than combine known techniques and apparatuses according to their known and ordinary uses to yield predictable results.

The Supreme Court further held that, "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a

different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill . . . ." *Id.* at 1740. Accordingly, a person of ordinary skill in the art at the time of the alleged invention would have been motivated to combine or adapt known or familiar methods in the art, especially where market forces prompt such variations. Here, market forces suggested a demand for some measure of accelerated loading of computers systems, and thus one of skill in the art would have thought to combine or modify references that described known methods that one of skill in the art would have recognized as offering improvements to solutions of that time. Each of the references describes methods that were known to offer such improvements, and, accordingly, one of skill in the art would have been motivated to combine or modify the references as identified in each of the combinations above.

Moreover, since there was a finite number of predictable solutions, a person of ordinary skill in the art had good reason to pursue the known options. *Id.* The above-identified prior art references merely use those familiar elements for their primary or well-known purposes in a manner well within the ordinary level of skill in the art. Accordingly, common sense and the knowledge of the prior art render the claims invalid under either Section 102 or Section 103.

A person of ordinary skill would have been motivated to combine the above prior art based on his knowledge, the nature of the problem to be solved, and the teachings of the prior art. The identified prior art addresses the same or similar technical issues and suggests the same or similar solutions to those issues. Moreover, some of the prior art refer to or discuss other prior art, illustrating the close technical relationship among the prior art.

By way of further example, the references listed in Section II above are directed to the same or similar technology. Thus, for example, one of ordinary skill in the art would have been motivated to combine known prior art solutions described in these references relating to the loading or boot operation of computer systems.

Moreover, as detailed below, many of the claim elements were already known as admitted by the applicants in the specifications of the Patents-in-Suit. These elements represented design choices available to a person of ordinary skill. When there is a design need or market pressure to solve a problem such as identified previously and/or described in the Patents-in-Suit, and there are a finite number of identified, predictable solutions, a person of ordinary skill would be motivated to combine the known options that are within his or her technical grasp. Here, one of ordinary skill in the art would have been motivated to combine known prior art solutions represented in the primary references. These motivations apply throughout each of the combinations below and in the claim charts.

In some of the paragraphs below, exemplary combinations are listed for purposes of explaining and exemplifying which references would be combined under the motivation cited in that particular paragraph. These combinations are not intended to be limiting but rather intended to provide notice of the reasoning behind particular motivations to combine and the references that would be combined thereunder. In many instances, multiple different motivations would apply to a particular combination (for example, if multiple references were both authored by the same person and referred to the same product/system, then both of those motivations would apply) and the specific combinations were not duplicated.

Should any prior art cited in Section II (and charted in Appendices A1-C38) be deemed not to disclose, explicitly or inherently, any limitation of a claim, Apple reserves the right to argue that

any such difference between that prior art and the corresponding claim would have been obvious to one of ordinary skill in the art. To the extent that such an argument is deemed a "combination" analysis for purposes of obviousness, Apple discloses its present intention to rely on the separate combination of the knowledge of a person of ordinary skill in the art with each item of prior art identified in Section II. The knowledge of a person of ordinary skill in the art is demonstrated by the full list of references in Section II and the prior art produced pursuant to Patent Rule 3-4(b).

For each of the various "systems" or "products" that were charted (*e.g.*, the Systems and Products), to the extent that the collective references cited in each Appendix are not considered to be the same reference for purposes of invalidity under 35 U.S.C. § 102, it would have been obvious to combine the references cited in each such Appendix. For each such product/system, all of the listed references relate to the same computer architecture and variations thereof. One of skill in the art would have been motivated to look at all of the available documentation for a particular product/system to understand the operation. One of skill in the art would have combined that knowledge from the various related references because it would be clear that the references were related. For at least this reason, one of skill in the art could consider it obvious to combine the references that collectively teach a particular operating system. The references were identified in the charts for each such system/product, and Apple intends to rely upon the combination of references in each of the system/product charts.

One of skill in the art would have been motivated to combine the different publications and patents that were authored by employees of the company or assigned to the same assignee and related to the same subject matter, particularly when that assignee was well-known to have experience and knowledge in the design and development of computer architectures and the patents are directed to improvements in the company's products. So, for example, for such

companies (*e.g.*, Apple, Toshiba Corporation, Hewlett-Packard Company, and/or International Business Machines, Inc., etc.) one of skill in the art would have been motivated to look at the various patents assigned to the companies, as well as articles written about their products, and to combine the approaches taken in those patents and articles for such products.

One of skill in the art would have been motivated to combine different references that were authored, developed, or invented by the same individual(s) related to the same subject matter. So, for example, individuals such as Burrows, Cheng, Wynn, Douglis, Eshfahini, Lillich, and others, were either the named author (e.g., for a paper) or a named inventor (on a patent) or known as a product architect on multiple related references charted in Appendices A1-C38. The common inventor/author/architect references themselves demonstrate that they relate to continued work in a common field of effort and continued related developments in that field. One of skill in the art would, therefore, combine the references related to each individual.

Based on the teachings of the references and/or the knowledge of one of ordinary skill, one of skill in the art would have been motivated to combine different references from the same company that relate to various generations of boot operations or data compression systems that were developed, made and sold by that same company—such as Apple or IBM. For example, the references and/or companies themselves described how the products evolved and were related to one another.

One of skill in the art would have been motivated to combine different references from boot operations and/or compression that were known to be related or subject to collaboration by different companies or individuals. It would have been obvious to combine references and aspects of these different but closely related disclosures. For example, it would have been obvious to combine the teachings of references related to the Linux OS (*e.g.*, Bennett, Grove, Rubini, and the

Linux Kernel) based on their closely related subject matter. By way of an additional example, it would have been obvious to combine the teachings of references related to input/output systems and/or the boot or load operation of such systems (*e.g.*, Anyimi, Hillis, Kikinis, Shipman, Teoman)[6] based on their closely related subject matter. In addition, it would have been obvious to one of skill in the art to apply the teachings of such input/output systems and/or the boot or load operations of such to other boot or load operations, such as those related to operating systems or applications, for the reasons discussed herein.

Each of the references can be combined with the known prior art based upon the statements made in the specification and during prosecution of the Patents-in-Suit. For example, the applicants acknowledged that many of the claimed features were well known in the art. *See* Section II.D, above. Thus, it would have been obvious for one of skill in the art to combine this knowledge in the art (as the applicants did) with any of the charted references because the applicants acknowledged that the basic arrangement of elements was known in the art.

Additionally, beyond the elements acknowledged in the patent and prosecution history to be known in the art, to the extent that the Realtime alleges that certain elements are missing from any of the charted references, those elements are known in the art or would be inherent in the relevant context. One of skill in the art would have understood that different approaches could be taken depending on the design needs and would have combined those approaches to meet the various design needs.

It would have been obvious to combine any references that expressly refer to each other or incorporate each other by reference. For example, one of skill in the art would have been motivated to combine different references from articles or papers that were known to be related

[6] *See also* U.S. Patent No. 6,564,318 to Gharda and U.S. Patent No. 6,567,911 to Mahmoud.

and/or referenced within another related article or paper. So, for example, it would have been obvious to combine a paper referenced in an article with the paper that references the article.

In addition to the specific combinations of prior art and the specific combinations of groups of prior art disclosed, Apple reserves the right to rely on any other combination of any prior art references disclosed in Section II above and charted in Appendices A1-C38. Apple further reserves the right to rely upon combinations disclosed within the prosecution history of the references cited herein. These obviousness combinations reflect Apple's present understanding of the potential scope of the claims that Realtime appears to be advocating and should not be seen as Apple's acquiescence to Realtime's interpretation of the patent claims.

In addition to the motivations set forth above, one or more combinations of the prior art references identified above and below pursuant to Patent Rule 3-3(a) would have been obvious because these references would have been combined using: known methods to yield predictable results; known techniques in the same way; a simple substitution of one known, equivalent element for another to obtain predictable results; and/or a teaching, suggestion, or motivation in the prior art generally. In addition, it would have been obvious to try combining the prior art references identified above pursuant to Patent Rule 3-3(a) because there were only a finite number of predictable solutions and/or because known work in one field of endeavor prompted variations based upon predictable design incentives and/or market forces either in the same field or a different one. In addition, the combinations of the prior art references identified above pursuant to P.R. 3-3(a) would have been obvious because the combinations represent the known potential options with a reasonable expectation of success.

Additional evidence that there would have been a motivation to combine the prior art references identified above pursuant to P.R. 3-3(a) includes the interrelated teachings of multiple

prior art references; the effects of demands known to the design community or present in the marketplace; the existence of a known problem for which there was an obvious solution encompassed by the claims of the Patents-in-Suit; the existence of a known need or problem in the field of the endeavor at the time of the alleged invention(s); and the background knowledge that would have been possessed by a person having ordinary skill in the art. For example, the prior art references are directed to solving the same problem: boot processes. Thus, a skilled artisan seeking to solve this problem of boot processes would have looked to these cited references alone or in combination. Accordingly, the motivation to combine the teachings of the prior art references disclosed herein is found in the references themselves and: (1) the nature of the problem being solved, (2) the express, implied and inherent teachings of the prior art, (3) the knowledge of persons of ordinary skill in the art, (4) the fact that the prior art is generally directed toward methods and systems for booting a computer system, or (5) the predictable results obtained in combining the different elements of the prior art.

Any reference or combination of references that anticipates or makes obvious an independent claim also makes obvious any claim dependent on that independent claim because every element of each dependent claim was known by a person of ordinary skill at the time of the alleged invention, and it would have been obvious to combine those known elements with the independent claim at least as a matter of common sense and routine innovation. Accordingly, Apple contends that each claim would have been obvious not only by the combinations explicitly defined in these contentions, but also by any combination of references that renders obvious that claim.

To the extent Realtime contends that any reference contains multiple distinct embodiments, it would be obvious to combine elements of the distinct embodiments. A person would be

motivated to make such a combination because the elements are found in the same reference and the reference as a whole is directed to the same topic or topics.

VII. ADDITIONAL BASES FOR OBVIOUSNESS UNDER 35 U.S.C. § 103

A. The State Of The Art

Numerous prior art references, including those identified above pursuant to P.R. 3-3(a), reflect common knowledge and the state, scope, and content of the prior art before the priority dates of the Patents-in-Suit. *See Graham v. John Deere Co.,* 383 U.S. 1, 35-36 (1966). As it would be unduly burdensome to create detailed claim charts for the thousands of invalidating combinations, the combinations cited herein are illustrative and not exhaustive. Though the claim charts provide illustrative citations to where each element may be found in the prior art references or in their combination, the cited references may contain other disclosures of each claim element as well, and Apple reserves the right to argue any claim elements of the asserted claims of the Patents-in-Suit are disclosed in non-cited portions of these references.

The state of the art included knowledge of the following (support for the following points below can be found in the art cited in Section II above, the charts cited in Section III above, and in Section VI above):

- What dictionary encoding is, how to implement various dictionary encoders and decoders, and the relative characteristics of various dictionary encoders and decoders;
    - That LZ77 is an adaptive dictionary encoder;
    - That LZ78 is an adaptive dictionary encoder;
    - That LZSS is an adaptive dictionary encoder;

- That many adaptive dictionary encoders are based on the fundamental work of Abraham Lempel and Jacob Ziv as described in Ziv, Jacob; Lempel, Abraham (May 1977). "A Universal Algorithm for Sequential Data Compression". IEEE Transactions on Information Theory 23 (3): 337–343; and Ziv, Jacob; Lempel, Abraham (September 1978). "Compression of Individual Sequences via Variable-Rate Coding". IEEE Transactions on Information Theory 24 (5): 530–536;
- The differences between various types of dictionary encoders, such as LZ77, LZ78, LZW, LZSS and others;
- The advantages and disadvantages associated with various dictionary encoders;
- That many dictionary encoders are adaptive and dynamically add newly encountered strings to the dictionary;
- That many dictionary encoders create a code word for each string added to a dictionary;
- That many dictionary encoders use indexes as code words;
- The types of data that are particularly suited for the various types of dictionary encoders;
- That LZ-based encoders are all based on the fundamental work of Abraham Lempel and Jacob Ziv as described in Ziv, Jacob; Lempel, Abraham (May 1977). "A Universal Algorithm for Sequential Data Compression". IEEE Transactions on Information Theory 23 (3): 337–343; and Ziv, Jacob; Lempel, Abraham (September 1978). "Compression of Individual

Sequences via Variable-Rate Coding". IEEE Transactions on Information Theory 24 (5): 530–536;

- The relative compression speed of various Lempel-Ziv encoders, such as LZ77, LZ78, LZW, LZSS and others;
- The relative decompression speed of various Lempel-Ziv decoders, such as LZ77, LZ78, LZW, LZSS and others;
- The relative compression efficiency of various Lempel-Ziv encoders, such as LZ77, LZ78, LZW, LZSS and others;
- The relative memory requirements of various Lempel-Ziv encoders, such as LZ77, LZ78, LZW, LZSS and others;
- The relative memory requirements of various Lempel-Ziv decoders, such as LZ77, LZ78, LZW, LZSS and others;
- That the various LZ-based encoders and features of LZ-based encoders can be readily combined, substituted and/or modified.

- What arithmetic and Huffman encoding and decoding was, how to implement arithmetic and Huffman encoding and decoding systems, and the relative characteristics of arithmetic and Huffman encoding and decoding systems;
  - The advantages and disadvantages of arithmetic and Huffman encoding
  - The relative compression speed of various Huffman encoders;
  - The relative decompression speed of various Huffman decoders;
  - The relative compression efficiency of various Huffman encoders;
  - The relative memory requirements of various Huffman encoders;
  - The relative memory requirements of various Huffman decoders;

- The types of data that are particularly suited for arithmetic and Huffman encoding.

- That there were a variety of additional encoding schemes and how to implement a variety of additional encoding schemes;
  - The advantages and disadvantages of each of a variety of encoding; and
  - The relative compression speed of various encoders;
  - The relative decompression speed of various decoders;
  - The relative compression efficiency of various encoders;
  - The relative memory requirements of various encoders;
  - The relative memory requirements of various decoders;
  - The types of data that are particularly suited each of a variety of encoding.
- That the use of different encoding schemes for different types of data could be beneficial.
- That the use of different encoding schemes for data that would be compressed once and decompressed many times could be beneficial.
- The relative design trade-offs between the following elements: compression speed, decompression speed, compression efficiency, compression memory requirements, decompression memory requirements, compression frequency, and decompression frequency.
  - The various circumstances under which decreased compression speed could be advantageously traded off against increased decompression speed; and

- The various circumstances under which decreased compression speed could be advantageously traded off against decreased decompression memory requirements;
- The various circumstances under which decreased compression speed could be advantageously traded off against increased decompression frequency; and
- The various circumstances under which decreased compression efficiency could be advantageously traded off against decreased decompression speed;
- The various circumstances under which decreased compression efficiency could be advantageously traded off against decreased decompression memory requirements;
- The various circumstances under which increased compression efficiency could be advantageously traded off against decreased decompression speed;
- The various circumstances under which increased compression efficiency could be advantageously traded off against decreased decompression memory requirements;
- The various circumstances under which increased compression memory requirements could be advantageously traded off against decreased compression speed;
- The various circumstances under which increased compression memory requirements could be advantageously traded off against increased compression efficiency;

- The various circumstances under which increased compression memory requirements could be advantageously traded off against decreased decompression memory requirements;
- The various circumstances where decreased compression efficiency could be advantageously traded off against decreased decompression memory requirements;
- The various circumstances where decreased compression efficiency could be advantageously traded off against increased decompression speed;
- The various circumstances where increased decompression memory requirements could be advantageously traded off against increased decompression speed; and
- Other advantageous tradeoffs between the above compression and decompression characteristics.

- That analyzing characteristics of the data to be compressed and how the compressed data will be used can allow the selection of an encoder and corresponding decoder that is well suited for compressing the data.
  - That compression of data can be optimized by selecting an encoder and corresponding decoder that is well-suited to the data.
- That a single compression system could use two or more encoding schemes to improve performance.
  - That run length compression and dictionary compression have been combined in several systems.
  - That compression schemes can be used in parallel.

- That compression schemes can be used in series.

- That many encoders receive uncompressed data and output compressed data.
  - That many encoders receive blocks of data.
  - That many encoders operate on blocks of data.
- That many encoders reduce repetition of data.
- That tags, or descriptors, can be used in compressed data to identify the compression scheme used to compress the data.
  - That a decompression system can use a tag or descriptor in compressed data to determine which decoder to use to decompress the data.
  - That tags or descriptors in compressed data can be stored with the compressed data.
- The various characteristics of caches were well-known:
  - That caches exist in many places in a computer system.
  - That caches are commonly placed in many places of the memory hierarchy including, but not limited, on a disk drive, on a disk controller, within the main memory subsystem, within the system's main processors.
  - That caches frequently have less capacity than the corresponding uncached element of the memory hierarchy.
  - That caches frequently have lower latency than the corresponding uncached element of the memory hierarchy.
  - That caches frequently have higher performance than the corresponding uncached element of the memory hierarchy.

- That caches frequently have higher performance than the corresponding uncached element of the memory hierarchy.
- That caches have been used at various places of the memory hierarchy for years prior to the filing of the Patents-in-Suit.

- The various characteristics of compressed and decompressed data were well-known:
  - That compressed data consumes less storage space than the corresponding uncompressed data.
  - That compressed data can be transmitted using less bandwidth than the corresponding uncompressed data.
  - That compressed data can be read from storage using less bandwidth than the corresponding uncompressed data.
  - That compressed data can be read from storage in less time than the corresponding uncompressed data.
  - That compressed data read from storage is read at a higher effective bandwidth than reading the corresponding uncompressed data from storage.
  - That compressed data can be read from storage using fewer processing resources than the corresponding uncompressed data.
  - That compressed data can be read from storage and stored in a cache in compressed form.
  - That compressed data can be read from storage and stored in a cache in decompressed form.

- That compressed data read from storage, stored in a cache, and subsequently read and decompressed from cache can be read from that cache at a higher effective bandwidth than reading the corresponding uncompressed data from storage.

- That compressed data can be stored more quickly than the corresponding uncompressed data.
  - That compressed data can be stored on a variety of storage media.
  - That compressed data can be stored at a higher effective bandwidth on a variety of storage media.
  - That stored compressed data can be retrieved from a variety of storage media.
  - That compressed data can be retrieved more quickly than the corresponding uncompressed data from a variety of storage media.
  - That compressed data can be retrieved at a higher effective bandwidth than the corresponding uncompressed data from a variety of storage media.
- How a storage device's bandwidth is determined and measured.
- The various characteristics of prefetching were well-known:
  - That prefetching frequently decreases latency to the prefetched entity when compared to fetching the corresponding entity without prefetching.
  - That prefetching frequently increases the apparent performance of the system when compared to fetching the corresponding entity without prefetching.

- That prefetching frequently increases the apparent bandwidth of the system when compared to fetching the corresponding entity without prefetching.
- That prefetching has been taught and/or used by various subsystems for years prior to the filing of the Patents-in-Suit.

- That various characteristics of storage access tracing were well-known:
  - That storage access tracing reveals access patterns and sequencing associated with a particular workload.
  - That storage access tracing can be performed by many elements in a computer system.
  - That storage access tracing can be performed by a computer system's storage controller.
  - That storage access tracing can be performed by a computer system's operating system.
  - That storage access tracing can be performed by a computer system's filesystem.
  - That storage access tracing reveals data that should be prefetched based on a high likelihood of subsequent use.
  - That storage access tracing reveals data that could be prefetched based on a high likelihood of subsequent use.
  - That storage access tracing reveals data that need not be prefetched based on a low likelihood of subsequent use.
  - That storage access tracing can be used to produce an enumeration of data to be prefetched based on a high likelihood of subsequent use.

- That storage access tracing can be used to produce a list of data to be prefetched based on a high likelihood of subsequent use.
- That the use of an enumeration of data to be prefetched could increase the likelihood that data was more readily available than if it had not been prefetched.
- That the use of a list of data to be prefetched could increase the likelihood that data was more readily available than if it had not been prefetched.
- That the refinement of an enumeration of data to be prefetched could increase the likelihood that data was more readily available than if the enumeration of data to be prefetched had not been refined.
- That the refinement of a list of data to be prefetched could increase the likelihood that data was more readily available than if the list of data to be prefetched had not been refined.
- That the refinement of an enumeration of data to be prefetched could include deleting an element from the enumeration of data that was not subsequently used after it had been prefetched.
- That the refinement of an enumeration of data to be prefetched could include adding an element to the enumeration of data that was not previously part of the enumeration of data to be prefetched but, nevertheless, was subsequently fetched and had not been prefetched.
- That the refinement of a list of data to be prefetched could include deleting an element from the list of data that was not subsequently used after it had been prefetched.

- That the refinement of a list of data to be prefetched could include adding an element to the list of data that was not previously on the list of data to be prefetched but, nevertheless, was subsequently fetched and had not been prefetched.
- That storage access tracing has been taught and/or used by various subsystems for years prior to the filing of the Patents-in-Suit.

- That various characteristics of prefetching data to accelerate operating system or program launch were well-known:
  - That prefetching data could be used to accelerate operating system or program launch.
  - That storage access tracing could be used to determine the appropriate data to be prefetched to accelerate operating system or program launch.
  - That storage access tracing could be used to produce an enumeration of data to be prefetched to accelerate operating system or program launch.
  - That storage access tracing could be used to produce a list of data to be prefetched to accelerate operating system or program launch.
  - That an enumeration of data could be further refined to accelerate operating system or program launch.
  - That a list of data could be further refined to accelerate operating system or program launch.
  - That accelerating operating system or program launch has been taught and/or used by various systems for years prior to the filing of the Patents-in-Suit.

- That various characteristics of compressing data in conjunction with operating system or program launch or initialization were well-known:
    - That compressing data used for an operating system or program would reduce the actual amount of required storage for a given operating system or program.
    - That compressing data used for an operating system or program would reduce the increase the effective amount of available storage for a given operating system or program.
    - That compressing data used for an operating system or program could reduce the amount of time required to read that compressed data from storage.
    - That compressing data used for an operating system or program could increase the effective read bandwidth to that compressed data.
    - That compressing data used for an operating system or program has been taught and/or used by various systems for years prior to the filing of the Patents-in-Suit.
- That various characteristics of decompressing compressed data in conjunction with operating system or program launch or initialization were well-known:
    - That decompressing compressed data in conjunction with operating system or program launch or initialization could reduce the effective amount of time required to read the corresponding uncompressed data.

- That decompressing compressed data in conjunction with operating system or program launch or initialization could increase the effective bandwidth to the corresponding uncompressed data.
- That decompressing compressed data in conjunction with operating system or program launch or initialization could result in a faster operating system or program launch.
- That decompressing compressed data in conjunction with operating system or program launch or initialization has been taught and/or used by various systems for years prior to the filing of the Patents-in-Suit.

- Techniques for creating a compressed kernel and booting a compressed kernel, including creating and booting a compressed Linux kernel were well-known.

B. The Admitted Prior Art

As discussed in Section II.D above, the patent applicants admitted in the specification of the Patents-in-Suit that many of the claimed features and techniques were well known in the prior art. Thus, Realtime is barred from denying the obviousness of those admittedly-old aspects of the claims.

C. Secondary Considerations

Notwithstanding the factors and motivations identified above, including the exemplary combinations identified above and in the claim charts of the Appendices, and notwithstanding the nascent stage of discovery, and subject to the reservation of rights stated above, Apple contends that an analysis of secondary considerations further supports the view that each of the asserted claims is obvious. Secondary considerations that courts evaluate as objective indicia of obviousness or nonobviousness of an alleged invention include (1) commercial success of the

claimed subject matter; (2) long felt but unresolved needs; (3) failure of others; (4) teaching away from the claimed subject matter by the prior art; (5) copying or acclamation by others; and (6) skepticism of experts. *See, e.g.*, *Ruiz v. A.B. Chance Co.,* 357 F.3d 1270, 1274 (Fed. Cir. 2004); *Ecolochem, Inc. v. Southern Cal. Edison Co.*, 227 F.3d 1361, 1379 (Fed. Cir. 2000).

Upon information and belief, neither Plaintiff, nor any predecessor in interest of the Patents-in-Suit has developed a commercially successful product embodying the claimed subject matter of these patents. The Patents-in-Suit also were not directed to long felt, unresolved needs. On the contrary, the Patents-in-Suit addressed problems that had been handled successfully in the prior art.

Additionally, and as outlined herein and in the Appendices, numerous prior art references anticipate the claims of the Patents-in-Suit, so failure by others cannot be cited as a secondary consideration in favor of nonobviousness. Numerous prior art references, including those identified above pursuant to Patent Rule 3-3(a), are directed to the same approach as the Patents-in-Suit regarding compression. Importantly, none of the prior art teaches away from the claimed subject matter. In fact, just the opposite is true, and as shown in these Invalidity Contentions and the accompanying claim charts, the prior art teaches the claimed subject matter. Thus, Realtime cannot rely upon any teaching away in the prior art.

Realtime has not presented any evidence to suggest that others in the industry copied or praised the alleged invention of the Patents-in-Suit. To the extent that others may have subsequently adopted a similar technique, Apple asserts that they were in fact copying well-known systems that predate the Patents-in-Suit. At the time of the alleged invention of the Patents-in-Suit, experts would not have been skeptical of the general approach or idea disclosed therein. The general idea—using multiple compression techniques—had already been in widespread use for a

long time and was not patentable by the time the applicants filed their patent applications. Experts would have regarded the disclosures and matter claimed in the Patents-in-Suit as obvious.

VIII. INVALIDITY UNDER 35 U.S.C. § 112(1/a)

Pursuant to Patent Rule 3-3(d), Apple hereby identifies grounds of invalidity based on lack of written description and enablement under 35 U.S.C. § 112(1/a) (requiring "a written description of the [claimed] invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same."). Apple reserves the right to supplement or amend these contentions based on Realtime's representations during the course of the litigation and any claim construction rulings of the Court.

Realtime has asserted in its Patent Rule 3-1(e) disclosures, for each Patent-in-Suit, that each asserted claim of the patent is entitled to the filing date of U.S. provisional application No. 60/180,114, filed on February 3, 2000, as its priority date (hereinafter, "the provisional application"). Apple denies that asserted entitlement but in view of such position of Realtime in its Patent Rule 3-1(e) disclosure, these invalidity contentions focus both on the lack of support in the provisional application for each claim and the lack of support in each non-provisional application for each claim.

Throughout these contentions, an assertion that an application did not "support" a claim or claim element means—unless otherwise noted—that the application did not provide as of the filing date sought:

(a) the required written description for the claim element and the claimed subject matter;

(b) the required written description for the full-scope of the claim element and the claim;

(c) the required enabling disclosure for the claim element and the claimed subject matter;

(d) the required enabling disclosure for the full-scope of the claim element and the claim;

(e) a description of the claimed "invention" understandable to a skilled artisan and showing that the applicant actually possessed such "invention";

(f) a description of the claimed "invention" that was express or necessarily present (inherent), complete, unambiguous, specific, and as broad as the claim; or

(g) a disclosure teaching a skilled artisan how, by following the steps set forth in the application, to make or carry out (use) the claimed "invention" without undue experimentation.

Throughout these contentions, an assertion that an application did not "support" an independent claim or language in an independent claim means—unless otherwise noted—that the application did not support that claim's asserted dependent claims either.

A. All Asserted Claims Are Invalid Under Sec. 112(1/a) In View of the Provisional Application

Each asserted claim of the asserted patents are invalid for failure of the provisional application to support the claim. For each asserted claims, the provisional application failed to support the claim and individual elements of the claim, as listed below.

Claim	Unsupported Claim or Individual Element
'608 Claim 1	• initializing a central processing unit of the computer system • preloading the boot data into cache memory prior to completion of initialization of the central processing unit of the computer system • preloading the boot data comprises accessing compressed boot data from a boot device • servicing requests for boot data from the computer system using the preloaded data after completion of initialization of the central processing unit of the computer system • accessing compressed boot data from the cache at a rate that increases the effective access rate of the cache
'608 Claim 4	• updating the list of boot data
'608 Claim 7	• for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system • and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system
'608 Claim 9	• preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device • servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises

Claim	Unsupported Claim or Individual Element
	accessing compressed application data from the cache and decompressing the compressed application data
’608 Claim 10	• further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device
’608 Claim 11	• wherein the decompressing is provided by a data compression engine
’608 Claim 12	• further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine
’608 Claim 15	• wherein Lempel-Ziv encoding is utilized to provide the compressed boot data
’608 Claim 16	• wherein a plurality of encoders are utilized to provide the compressed boot data
’608 Claim 19	• wherein Lempel-Ziv encoding is utilized to provide the compressed boot data
’608 Claim 20	• wherein a plurality of encoders are utilized to provide the compressed boot data
’608 Claim 22	• initializing a central processing unit of the computer system • preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit • servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data • with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device
’608 Claim 24	• wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data
’608 Claim 25	• wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data
’608 Claim 27	• preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system • servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit • a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device
’608 Claim 29	• wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data
’608 Claim 30	• wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data
’936 Claim 1	• initializing a central processing unit of said computer system • at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device • preloading said at least a portion of said boot data in compressed form from said boot device into memory; • accessing and decompressing said at least a portion of said boot data in said compressed for from said memory • utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine

Claim	Unsupported Claim or Individual Element
'936 Claim 2	• wherein said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system
'936 Claim 3	• wherein said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system
'936 Claim 4	• wherein said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system
'936 Claim 6	• updating the list of boot data
'936 Claim 8	• wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form
'936 Claim 9	• wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form
'936 Claim 11	• said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system • a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data
'936 Claim 15	• wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form
'936 Claim 16	• wherein a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form
'862 Claim 1	• loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory; • accessing the loaded portion of the boot data in the compressed form from memory • decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form • updating the boot data list • wherein the decompressed portion of boot data comprises a portion of the operating system
'862 Claim 2	• wherein the updating comprises: associating additional boot data with the boot data list
'862 Claim 3	• wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list
'862 Claim 4	• wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data
'862 Claim 5	• storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory • loading the stored compressed boot data from the first memory; • utilizing the decompressed boot data to at least partially boot the computer system; • updating the boot data list • wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form
'862 Claim 6	• a memory • to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory • to access the loaded portion of the boot data in the compressed form

Claim	Unsupported Claim or Individual Element
	• to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data • to update the boot data list
’862 Claim 8	• loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list • storing a portion of the operating system in a compressed form in a first memory; • utilizing the decompressed portion of the operating system to at least partially boot the computer system • updating the boot data list • wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form
’862 Claim 9	• compressing an additional portion of the operating system that is not associated with the boot data list • utilizing the stored additional portion of the operating system to at least further partially boot the computer system • storing the additional portion of the operating system in the first memory
’862 Claim 10	• storing the updated boot list in a non-volatile memory
’862 Claim 11	• loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system • decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form • updating the boot data list
’862 Claim 13	• loading boot data in a compressed form that is associated with a boot data list from a boot device • decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form • updating the boot data list
’862 Claim 14	• loading the boot date into a memory • accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list • servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form • updating the boot data list
’862 Claim 15	• wherein the boot data comprises: a program code associated with the operating system
’862 Claim 16	• wherein the operating system comprises: a plurality of files
’862 Claim 17	• a program code associated with the operating system and an application program
’862 Claim 19	• wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list • associating the accessed boot data that is not associated with the boot data list to the boot data list • and wherein the updating comprises associating the accessed boot data that is not associated with the boot data list to the boot data list

Claim	Unsupported Claim or Individual Element
’862 Claim 23	• wherein the portion of the boot data in the compressed form represents a plurality of files
’862 Claim 24	• wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system
’862 Claim 28	• wherein the operating system comprises: a plurality of files
’862 Claim 29	• a program code associated with the operating system and an application program
’862 Claim 31	• wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access
’862 Claim 32	• wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form
’862 Claim 33	• wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form
’862 Claim 35	• wherein the compressed boot data represents a plurality of files
’862 Claim 36	• a program code associated with an operating system of the computer system
’862 Claim 40	• wherein the operating system comprises: a plurality of files
’862 Claim 44	• wherein a form of dictionary encoding was utilized to encode the compressed boot data
’862 Claim 45	• wherein Lempel-Ziv encoding was utilized to encode the compressed boot data
’862 Claim 47	• wherein the boot data in the compressed form represents a plurality of files
’862 Claim 48	• a program code associated with an operating system
’862 Claim 52	• wherein the boot data the compressed form comprises: a plurality of files
’862 Claim 53	• a program code associated with an operating system of the system and an application program
’862 Claim 59	• wherein the operating system in the compressed form represents a plurality of files
’862 Claim 60	• program code associated with the operating system
’862 Claim 64	• a plurality of files
’862 Claim 65	• a program code associated with the operating system and an application program
’862 Claim 71	• wherein the boot data in the compressed form represents a plurality of files
’862 Claim 72	• a program code associated with the operating system
’862 Claim 76	• wherein the operating system comprises: a plurality of files
’862 Claim 77	• a program code associated with the operating system and an application program
’862 Claim 80	• wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form
’862 Claim 81	• wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form
’862 Claim 83	• wherein the boot data in the compressed form represents a plurality of files
’862 Claim 84	• a program code associated with the operating system
’862 Claim 88	• wherein the operating system comprises: a plurality of files
’862 Claim 89	• a program code associated with the operating system and application program
’862 Claim 92	• wherein a form of dictionary encoding was utilized to encode the compressed boot data
’862 Claim 93	• wherein Lempel-Ziv encoding was utilized to encode the compressed boot data

Claim	Unsupported Claim or Individual Element
'862 Claim 97	• accessing additional compressed boot data that is not associated with the boot data list • wherein the updating comprises: associating the additional compressed boot data with the boot data list
'862 Claim 98	• wherein the updating comprises: disassociating non accessed boot data from the boot data list
'862 Claim 107	• storing the updated boot list in a non-volatile memory
'862 Claim 109	• storing the compressed additional boot data

In addition, all of the asserted patents claim priority to the same provisional application. That provisional patent, however, unequivocally states that Realtime's proprietary data compression and encryption engine combined with their implementation of dedicated hardware and an ultra-high speed digital signal processor provides Realtime's alleged benefits and avoids the traditional delays associated with software data compression to provide an alleged improvement over the prior art. *See* Provisional Application at Preface ("In order to achieve this level of performance our proprietary data compression and encryption engine reads one byte of data stored on disk and decodes this information into multiple bytes of information for the computer. By implementing this process in a combination of dedicated hardware and an ultra high speed digital signal processor the translation takes place in 'real-time.' Thus, rather than the traditional delays normally associated with software data compression, our hardware approach creates a many-fold performance improvement."). The limitation of the alleged invention to dedicated hardware is further evidenced by Realtime's statement that its "technology is designed to be *Scalable* through each successive computer generation; *Adaptable* to serve multiple functions concurrently, increasing performance and enhancing value; *Insertable* to seamlessly integrate into existing marketplaces—modifications to existing standards are not required." *Id.* Accordingly, the provisional application expressly disclaims software only implementation of the asserted claims and any such implementation within the claims is unsupported by the provisional application.

B. All '608 Asserted Claims Are Invalid Under Sec. 112(1/a)

Each asserted claim of the '608 patent is invalid for failure of the '608 patent's application (application number 09/776,267, filed on February 2, 2001) to support the claim. For each asserted claim of the '608 patent, the application failed to support the claim and individual elements of the claim, as listed below.

Claim	Unsupported Claim or Individual Element
'608 Claim 1	• initializing a central processing unit of the computer system • preloading the boot data into cache memory prior to completion of initialization of the central processing unit of the computer system; and • servicing requests for boot data from the computer system using the preloaded data after completion of initialization of the central processing unit of the computer system; and • accessing compressed boot data from the cache at a rate that increases the effective access rate of the cache
'608 Claim 4	• updating the list of boot data
'608 Claim 7	• for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system; and • and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system
'608 Claim 9	• preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and • servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data
'608 Claim 10	• further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device
'608 Claim 11	• wherein the decompressing is provided by a data compression engine
'608 Claim 12	• further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine
'608 Claim 15	• wherein Lempel-Ziv encoding is utilized to provide the compressed boot data
'608 Claim 16	• wherein a plurality of encoders are utilized to provide the compressed boot data
'608 Claim 19	• wherein Lempel-Ziv encoding is utilized to provide the compressed boot data
'608 Claim 20	• wherein a plurality of encoders are utilized to provide the compressed boot data
'608 Claim 22	• initializing a central processing unit of the computer system • preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit

Claim	Unsupported Claim or Individual Element
	• servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data; and • with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device
'608 Claim 24	• wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data
'608 Claim 25	• wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data
'608 Claim 27	• preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system • servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit • a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device
'608 Claim 29	• wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data
'608 Claim 30	• wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data

C. All '936 Asserted Claims Are Invalid Under Sec. 112(1/a)

Each asserted claim of the '936 patent is invalid for failure of the '936 patent's application (application number 11/551,204, filed on October 19, 2006) to support the claim.[7] For each asserted claim of the '936 patent, the application failed to support the claim and individual elements of the claim, as listed below.

Claim	Unsupported Claim or Individual Element
'936 Claim 1	• initializing a central processing unit of said computer system • at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device; and • utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine
'936 Claim 2	• wherein said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system
'936 Claim 3	• wherein said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system

[7] In addition, because the Patents-in-Suit share a common specification, the predecessor application for the '608 patent (09/776,267) also does not support the asserted claims of the '936 patent.

Claim	Unsupported Claim or Individual Element
'936 Claim 4	• wherein said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system
'936 Claim 6	• updating the list of boot data
'936 Claim 8	• wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form
'936 Claim 9	• wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form
'936 Claim 11	• said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system • a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data
'936 Claim 15	• wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form
'936 Claim 16	• wherein a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form

D. All '862 Asserted Claims Are Invalid Under Sec. 112(1/a)

Each asserted claim of the '862 patent is invalid for failure of the '862 patent's application (application number 09/776,267, filed on May 27, 2011) to support the claim.[8] For each asserted claim of the '862 patent, the application failed to support the claim and individual elements of the claim, as listed below.

Claim	Unsupported Claim or Individual Element
'862 Claim 1	• loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory • accessing the loaded portion of the boot data in the compressed form from memory • decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form • wherein the decompressed portion of boot data comprises a portion of the operating system
'862 Claim 2	• associating additional boot data with the boot data list
'862 Claim 5	• storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory • utilizing the decompressed boot data to at least partially boot the computer system • wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form

[8] In addition, because the Patents-in-Suit share a common specification, the predecessor applications for the '608 patent (09/776,267) and U.S. Patent No. 8,112,619 (application no. 11/551,211) also do not support the asserted claims of the '862 patent.

Claim	Unsupported Claim or Individual Element
’862 Claim 6	• to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory • to access the loaded portion of the boot data in the compressed form • to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data
’862 Claim 8	• loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list • storing a portion of the operating system in a compressed form in a first memory • utilizing the decompressed portion of the operating system to at least partially boot the computer system • wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form
’862 Claim 9	• compressing an additional portion of the operating system that is not associated with the boot data list • utilizing the stored additional portion of the operating system to at least further partially boot the computer system • storing the additional portion of the operating system in the first memory
’862 Claim 10	• storing the updated boot list in a non-volatile memory
’862 Claim 11	• loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system • decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form
’862 Claim 13	• loading boot data in a compressed form that is associated with a boot data list from a boot device • decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form
’862 Claim 14	• accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list • servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form
’862 Claim 15	• wherein the boot data comprises: a program code associated with the operating system
’862 Claim 16	• wherein the operating system comprises: a plurality of files
’862 Claim 17	• a program code associated with the operating system and an application program
’862 Claim 19	• wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list • associating the accessed boot data that is not associated with the boot data list to the boot data list.
’862 Claim 23	• wherein the portion of the boot data in the compressed form represents a plurality of files
’862 Claim 24	• wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system
’862 Claim 28	• wherein the operating system comprises: a plurality of files

Claim	Unsupported Claim or Individual Element
'862 Claim 29	• a program code associated with the operating system and an application program
'862 Claim 31	• wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access
'862 Claim 32	• wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form
'862 Claim 33	• wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form
'862 Claim 35	• wherein the compressed boot data represents a plurality of files
'862 Claim 36	• a program code associated with an operating system of the computer system
'862 Claim 40	• wherein the operating system comprises: a plurality of files
'862 Claim 44	• wherein a form of dictionary encoding was utilized to encode the compressed boot data
'862 Claim 45	• wherein Lempel-Ziv encoding was utilized to encode the compressed boot data
'862 Claim 47	• wherein the boot data in the compressed form represents a plurality of files
'862 Claim 48	• a program code associated with an operating system
'862 Claim 52	• wherein the boot data the compressed form comprises: a plurality of files
'862 Claim 53	• a program code associated with an operating system of the system and an application program
'862 Claim 59	• wherein the operating system in the compressed form represents a plurality of files
'862 Claim 60	• program code associated with the operating system
'862 Claim 64	• a plurality of files
'862 Claim 65	• a program code associated with the operating system and an application program
'862 Claim 71	• wherein the boot data in the compressed form represents a plurality of files
'862 Claim 72	• a program code associated with the operating system
'862 Claim 76	• wherein the operating system comprises: a plurality of files
'862 Claim 77	• a program code associated with the operating system and an application program
'862 Claim 80	• wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form
'862 Claim 81	• wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form
'862 Claim 83	• wherein the boot data in the compressed form represents a plurality of files
'862 Claim 84	• a program code associated with the operating system
'862 Claim 88	• wherein the operating system comprises: a plurality of files
'862 Claim 89	• a program code associated with the operating system and application program

IX. INVALIDITY UNDER 35 U.S.C. § 112(2/b)

Pursuant to Patent Rule 3-3(d), Apple hereby identifies grounds of invalidity "based on indefiniteness" under 35 U.S.C. § 112(2/b) (requiring "claims particularly pointing out and distinctly claiming the" "invention"). Apple reserves the right to supplement or amend these

contentions based on Realtime's representations during the course of the litigation and any claim construction rulings of the Court. The Patent Rules do not require these contentions to include, and Apple does not include, contentions regarding invalidity under the additional "regards as the invention" requirement of Section 112(2/b).

Throughout these contentions, an assertion that a claim or claim language was unclear and imprecise means—unless otherwise noted—that the claim, each claim containing that claim language, and each dependent claim therefrom, read by a skilled artisan at the time of the patent application, in light of the application and its Patent Office prosecution history leading to issuance of the patent:

(a) failed the statute's particular and distinct claiming mandate;

(b) failed to inform, with reasonable certainty, those skilled in the art at the time of the patent application of the scope of the claimed invention; and

(c) failed to clearly distinguish what is claimed from what went before in the art and clearly circumscribe what is foreclosed from future enterprise.

An assertion that a claim or claim language was unclear and imprecise does not necessarily mean that the claim or claim language is not now amenable to construction by the court.

A. <u>All '608 Asserted Claims Are Invalid Under Sec. 112(2/b)</u>

All asserted claims of the '608 patent are unclear and imprecise on account of its language: "prior to completion of the initialization of the central processing unit" and/or "after completion of initialization of the central processing unit" in claims 1, 7, 9, 22, 27 (as well as those claims that depend from the same). For example, it is unclear and imprecise as to what qualifies or does not qualify as a "completion of the initialization" of a central processing unit.

In addition, claims 1 and 7 of the '608 patent (and those claims that depend from the same) are unclear and imprecise on account of the claim language "a rate that increases the effective access rate of the cache." For example, this claim language is unclear and imprecise as to a definition of the "effective access rate of the cache," the degree (if any) to which that rate is increased, and how (e.g., what materials, acts and structures) these claimed results are achieved. In addition, claim 1 is unclear and imprecise on whether this claim language is comparative and, if so, it fails to disclose or provide any guidance regarding the appropriate measure for any such comparison.

In addition, claims 1, 7, 9, 22, and 27 (and those claims that depend from the same), as well as claims 3 and 8, are unclear and imprecise on account of the claim language "preloading." For example, this claim language is unclear and imprecise as the distinction between preloading as opposed to loading. It is unclear and imprecise as to whether preloading prior to completion of the initialization of the central processing unit is distinct from loading prior to completion of the initialization of the central processing unit; or whether if no such distinction exists, the claim language "preloading" renders other claim language—namely, "prior to" redundant or superfluous.

In addition, claims 7 and 27 of the '608 patent (and those claims that depend from the same) are unclear and imprecise on account of the claim language "a non-volatile memory device, for storing logic code associated with the DSP or controller" and "non-volatile memory for storing logic code for use by the processor," respectively. For example, this claim language uses a generic placeholder ("non-volatile memory device for") and should be construed as a means-plus-function term, which lacks supporting structure in the specification.

In addition, claims 7 and 8 (and those claims that depend from the same) are unclear and imprecise on account of the claim language "instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system" and "instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system, for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system, and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system," respectively. For example, this claim language uses a generic placeholder ("instructions executable by the DSP or controller for") and should be construed as a means-plus-function term, which lacks supporting structure in the specification.

B. All '936 Asserted Claims Are Invalid Under Sec. 112(2/b)

All asserted claims of the '936 patent are unclear and imprecise on account of its language "preloading said at least a portion of said boot data in compressed form from said boot device to a memory . . . utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine" and "said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system," which appear in independent claims 1 and 11. For example, it is unclear to a person of ordinary skill how large said portion must be to infringe these claims.

In addition, claims 1 and 5 (and those claims that depend from the same), are unclear and imprecise on account of the claim language "preloading." For example, this claim language is

unclear and imprecise as the distinction between preloading as opposed to loading. It is unclear and imprecise as to whether preloading prior to completion of the initialization of the central processing unit is distinct from loading prior to completion of the initialization of the central processing unit; or whether if no such distinction exists, the claim language "preloading" renders other claim language—namely, "prior to" redundant or superfluous.

C. <u>All '862 Asserted Claims Are Invalid Under Sec. 112(2/b)</u>

Claim 1 of the '862 patent (and all claims that depend from the same) is unclear and imprecise on account the claim language "a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form." In addition, claims 5, 6, 8, 11, 13, and 14 (and all claims that depend from the same) are unclear and imprecise on account of similar claim language and limitation. For example, this claim language was unclear and imprecise on what supposedly occurs faster than what, and how (*e.g.*, what materials, acts and structures) these claimed results are achieved. For example, claim 1 is unclear and imprecise on whether this claim language accounts for the total time period taken to boot the entire computer system, or instead the time to load the portion of the boot data that is compressed.

In addition, claims 1, 6, 14 (and those claims that depend from the same), as well as claims 23, 24, 31, 32, and 33, are unclear and imprecise on account of the claim language "portion of boot data" or "portion of the boot data." For example, it is unclear to a person of ordinary skill how large said portion must be to infringe these claims.

In addition, claim 1 (and those claims that depend from the same), is unclear and imprecise on account of the claim language "wherein the decompressed portion of boot data comprises a portion of the operating system." For example, it is unclear and imprecise as to whether the

decompressed portion of the boot data is used to boot the operating system, or whether the decompressed portion of the boot data is a portion of the operating system itself.

In addition, claim 6 (and those claims that depend from the same) are unclear and imprecise on account of the claim language "a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor." For example, this claim language uses a generic placeholder ("a second memory configured to") and should be construed as a means-plus-function term, which lacks supporting structure in the specification. In addition, claim 6 is unclear and imprecise on account of the claim language "wherein the processor is configured to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory, to access the loaded portion of the boot data in the compressed form, to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data, and to update the boot data list." For example, this claim language uses a generic placeholder ("the processor is configured to") and should be construed as a means-plus-function term, which lacks supporting structure in the specification.

In addition, claims 5, 8, and 9 (and those claims that depend from the same), are unclear and imprecise on account of the claim language "partially boot the computer system." For example, it is unclear to a person of ordinary skill when it means to partially boot a computer system, or when or at what point a partial boot is complete.

X. ACCOMPANYING DOCUMENT PRODUCTION

Pursuant to Patent Rule 3-4(b), Apple is concurrently producing copies of the identified prior art as APL-REAL_00548259-APL-REAL_00552119. In addition, upon reasonable notice from Realtime, Apple will make available for inspection at the Dallas of Fish & Richardson P.C.

(or at another agreed upon office of Fish & Richardson, P.C.) source code related to the New World Mac System.

Apple's investigation is ongoing. If and when any additional documents required to be disclosed under this rule are discovered, Apple will timely produce them.

Dated: November 7, 2016

FISH & RICHARDSON P.C.

By: */s/ Michael A. Bittner*
Michael A. Bittner

Attorneys for Defendant
APPLE INC.

PROOF OF SERVICE

I am employed in the County of San Mateo. My business address is Fish & Richardson P.C., 500 Arguello Street, Suite 500, Redwood City, California 94063. I am over the age of 18 and not a party to the foregoing action.

I am readily familiar with the business practice at my place of business for collection and processing of correspondence for personal delivery, for mailing with United States Postal Service, for facsimile, and for overnight delivery by Federal Express, Express Mail, or other overnight service.

On November 7, 2016, I caused a copy of the following document(s):

DEFENDANT APPLE INC.'S INVALIDITY CONTENTIONS

to be served on the interested parties in this action by placing a true and correct copy thereof, enclosed in a sealed envelope, and addressed as follows:

David M. Saunders
David.Saunders@fischllp.com
Fisch Sigler LLP
96 North Third Street, Suite 260
San Jose, CA 95112
Telephone: 650.362.8200

Alan M. Fisch
Alan.Fisch@fischllp.com
R. William Sigler
Bill.Sigler@fischllp.com
Jeffrey M. Saltman
Jeffrey.Saltman@fischllp.com
Fisch Sigler LLP
5301 Wisconsin Avenue NW, Fourth Floor
Washington, DC 20015
Telephone: 202.362.3500

XX **ELECTRONIC MAIL:** Such document was transmitted by electronic mail to the addressees' email addresses as stated above.

I declare that I am employed in the office of a member of the bar of this Court at whose direction the service was made.

I declare under penalty of perjury that the above is true and correct. Executed on November 7, 2016, at Redwood City, CA.

/s/ Trevor Goodrich
Trevor Goodrich

Appendix A1
Invalidity of U.S. Patent 7,181,608 based on Esfahani

U.S. Patent Nos. 6,434,695 to Esfahani ("Esfahani '695) and 6,732,265 to Esfahani ("Esfahani '265"), alone or in combination, invalidate claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

In addition, the prior art New World Mac operating system disclosed in Esfahani ("New World Mac System") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of the '608 Patent pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 because the system was on sale or in public use more than one year prior to the filing date of the '608 patent. Esfahani '695, Esfahani '265, and the New World Mac System are collectively referred to herein as "Esfahani."

Additionally, Apple intends to provide and seek discovery related to this system to obtain copies of relevant materials, including but not limited to, architectural documents, design documents, implementation documents, internal publications, patent applications and business records relating to this product and will supplement these contentions after those materials are discovered or received. Apple also reserves its rights to rely on any additional New World Mac System materials, either individually or collectively, as prior art publications to the '608 patent.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A1
Invalidity of U.S. Patent 7,181,608 based on Esfahani

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Esfahani, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> The low-level portion, including hardware-specific code, is stored in a relatively small read-only memory (ROM), while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device, which may be located remotely from the computer system. Upon power-up or reset of the computer system, the code in the ROM is executed to read the compressed ROM image into random access memory (RAM) of the computer system. The compressed image is then decompressed and executed as part of the boot sequence. Once decompressed, the portion of RAM storing the intermediate-level code is write-protected in the memory map, and the code in boot ROM is deleted from the memory map. Memory space in RAM that is allocated to the intermediate-level code but not used is returned to the operating system for use as part of system RAM.

Esfahani '695 [Abstract]

> A method and apparatus for use in booting a computer system are provided. The method includes loading a compressed image of a first portion of the OS of the computer system into a storage device of the computer system, which may be RAM, for example. The compressed image of the first portion of the OS is then decompressed and executed as part of the boot sequence of the computer system.

Esfahani '695 2:6-12; *also* Esfahani '265, 2:10-17.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

1.1 maintaining a list of boot data used for booting a computer system;	Esfahani, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> During start-up, the Boot Info file 40 is loaded into RAM 12, and the compressed mid-level OS 32 is decompressed. Hence, the mid-level OS 32 is essentially a compressed ROM image. The mid-level OS 32 is inserted into the memory map of the computer system as if it were firmware in ROM. That is, the ROM image can be write-protected in the memory map.

Esfahani '695, 5:44-51; *also* Esfahani '265, 5:48-54.

> Referring now to FIG. 5, the low-level OS portion 31 and the Boot Info file 40 are illustrated in greater detail. The low-level portion 31 stored in Boot ROM 11 contains the code needed to start up the computer, initialize and examine the hardware, provide a Device Tree (per Open Firmware) to describe the hardware, provide hardware access services, and transfer control to the OS. In one embodiment, the components of the low-level portion 31 include:
>
> > code for performing Power-On Self Test (POST), including code for performing diagnostics, generating a boot beep and an error beep;
> >
> > Open Firmware code;
> >
> > hardware-specific Mac OS drivers ("ndrv's") that are needed at boot time (drivers needed at boot time, e.g.,video drivers, network drivers, or disk drivers, are loaded from the Device Tree);
> >
> > HardwareInit code (i.e., the lowest-level code for initializing the CPU, RAM, clock, system bus, etc.) without Mac OS-specific code;

> code for performing Run-Time Abstraction Services (RTAS). Certain hardware devices differ from machine to machine, but provide similar functions. RTAS provides such functions, including functions for accessing the real-time clock, NVRAM 20, restart, shutdown, and PCI configuration cycles. The I/O primitives for these functions in the ROM Image make use of RTAS.

Esfahani '695, 6:29-54; *also* Esfahani '265, 6:31-57.

> The Boot Info file 40 will now be described in greater detail. The Boot Info file 40 may be stored in the System Folder of the start-up volume. Alternatively, the Boot Info file 40 may be provided by a network server using, for example, the Bootstrap Protocol (BootP), which is described in B. Croft et al., Network Working Group Request for Comments (RFC) 951, September 1985. Referring to FIG. 5, the Boot Info file 40 includes Open Firmware-specific Mac OS code 52, referred to as the "Trampoline code"; an Open Firmware header 51, which includes a Forth script that performs operations necessary to the start-up of the OS, including validation tests and transfer of control to the Trampoline code; and a compressed ROM Image 53, which represents the mid-level portion 32 of the OS 30. The purpose of the header 51 is, generally, to specify the locations of the other components of the Boot Info file 40. The purpose of the Trampoline code 53 generally is to handle the transition between the Open Firmware code in the Boot Rom 11 and the ROM Image 52, as will be described in greater detail below.

Esfahani '695, 7:5-24; *also* Esfahani '265, 7:9:28.

> The Boot Info file 40 resides on the boot device (e.g., a disk, or on a network) and has a localizable name.

Esfahani '695, 7:66-67; *also* Esfahani '265, 8:3-4.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

1.2 initializing a central processing unit of the computer system;	Esfahani, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> Referring now to FIG. 5, the low-level OS portion 31 and the Boot Info file 40 are illustrated in greater detail. The low-level portion 31 stored in Boot ROM 11 contains the code needed to start up the computer, initialize and examine the hardware, provide a Device Tree (per Open Firmware) to describe the hardware, provide hardware access services, and transfer control to the OS. In one embodiment, the components of the low-level portion 31 include:
>
> > code for performing Power-On Self Test (POST), including code for performing diagnostics, generating a boot beep and an error beep;
> >
> > Open Firmware code;
> >
> > hardware-specific Mac OS drivers ("ndrv's") that are needed at boot time (drivers needed at boot time, e.g.,video drivers, network drivers, or disk drivers, are loaded from the Device Tree);
> >
> > HardwareInit code (i.e., the lowest-level code for initializing the CPU, RAM, clock, system bus, etc.) without Mac OS-specific code;
> >
> > code for performing Run-Time Abstraction Services (RTAS). Certain hardware devices differ from machine to machine, but provide similar functions. RTAS provides such functions, including functions for accessing the real-time clock, NVRAM 20, restart, shutdown, and PCI configuration cycles. The I/O primitives for these functions in the ROM Image make use of RTAS.

Esfahani ’695, 6:29-54; *also* Esfahani ’265, 6:31-57.

> Overall Boot Process
>
> Referring now to FIGS. 5, 6A and 6B, the overall boot process of the improved OS 30 will now be described. In response to power on or reset at block 601 (FIG. 6A), the POST code executes (preliminary diagnostics, boot beep, initialization) at 602. At 603, the Open Firmware routine initializes and begins execution, including building the expanded Device Tree and the Interrupt Trees. At 604, the Open Firmware locates the Boot Info file 40, based on defaults and NVRAM settings, and loads the Boot Info file 40 into RAM 12. At 605, the Open Firmware executes the Forth script in the Boot Info file 40, which contains information about the rest of the file (offsets, etc.) and instructions to read both the Trampoline code 52 and the compressed ROM Image 53, and places them into a temporary place in RAM 12. At 606, the Open Firmware transfers control to the Trampoline code 52. At 607, the Trampoline code 52 decompresses the ROM Image 53 in RAM 12. In addition, the Trampoline code reallocates any unused memory space in the ROM Image 52; gathers information about the system from Open Firmware; creates data structures based on this information; terminates Open Firmware, and rearranges the contents of memory to an interim location in physical memory space. At 608, the Trampoline code 52 transfers control to the HardwareInit routine of the (now decompressed) ROM Image 53. At 609, the HardwareInit routine copies data structures to their correct places in memory and then calls the kernel of the OS. Next, at 610 the kernel fills in its data structures and then calls the 68K Emulator. At 611, the 68K Emulator initializes itself and then transfers control to the StartInit routine. At 612, the StartInit routine begins execution, initializing data structures and managers, and booting the MacOS (the high-level portion of the OS). Note that blocks 609 through 612, above, are specific to a Mac OS.

Esfahani ’695, 9:38-10:6; *also* Esfahani ’265, 8:42-9:10.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Esfahani, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani 695 2:13-25; *also* Esfahani '265, 2:18-29.

> A computer OS using a compressed ROM image in RAM is described. In brief, the low-level portion of an OS of a computer is designed to be separate from the intermediate-level portion of the OS. The low-level portion, which includes hardware-specific code, is stored in a relatively small Boot ROM, while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device. The mass storage device may be located remotely from the computer system, such as in a file server. Upon power-up or reset of the computer system, the code in the boot ROM is executed to read the compressed ROM image into RAM, i.e., system memory, of the computer system. The compressed image is then decompressed and executed as part of the boot sequence.

Esfahani 695, 2:54-67; *also* Esfahani '265, 2:58-3:4.

> Referring now to FIG. 4, in the improved OS, the low-level (hardware-specific) OS code 31 resides in firmware, in order to handle start-up activities of the computer system. This code fits into one, relatively small ROM, referred to as the Boot ROM 11. Thus, Boot ROM 11 includes all of the hardware-specific code and tables needed to start up the computer as well as to boot the OS and provide common hardware access services the OS might require. Note that the Boot ROM code is not specific to the MacOS or to any other OS. All higher-level software resides elsewhere, as will now be described.
>
> Prior to start-up, the mid-level portion 32 of OS 30 (which corresponds to part of the ToolBox ROM of earlier Macintosh computers) resides in compressed form in a file 40, referred to as Boot Info file 40.

Esfahani '695, 5:7-23; *also* Esfahani '265, 5:10-24.

> During start-up, the Boot Info file 40 is loaded into RAM 12, and the compressed mid-level OS 32 is decompressed. Hence, the mid-level OS 32 is essentially a compressed ROM image. The mid-level OS 32 is inserted into the memory map of the computer system as if it were firmware in ROM. That is, the ROM image can be write-protected in the memory map.

Esfahani '695, 5:44-51; *also* Esfahani '265, 5:48-54.

> Referring now to FIG. 5, the low-level OS portion 31 and the Boot Info file 40 are illustrated in greater detail. The low-level portion 31 stored in Boot ROM 11 contains the code needed to start up the computer, initialize and examine the hardware, provide a Device Tree (per Open Firmware) to describe the hardware, provide hardware access services, and transfer control to the OS. In one embodiment, the components of the low-level portion 31 include:
>
> > code for performing Power-On Self Test (POST), including code for performing diagnostics, generating a boot beep and an error beep;
> >
> > Open Firmware code;
> >
> > hardware-specific Mac OS drivers ("ndrv's") that are needed at boot time (drivers needed at boot time, e.g.,video drivers,

> network drivers, or disk drivers, are loaded from the Device Tree);
>
> HardwareInit code (i.e., the lowest-level code for initializing the CPU, RAM, clock, system bus, etc.) without Mac OS-specific code;
>
> code for performing Run-Time Abstraction Services (RTAS). Certain hardware devices differ from machine to machine, but provide similar functions. RTAS provides such functions, including functions for accessing the real-time clock, NVRAM 20, restart, shutdown, and PCI configuration cycles. The I/O primitives for these functions in the ROM Image make use of RTAS.

Esfahani ’695, 6:29-54; *also* Esfahani ’265, 6:31-57.

> The Boot Info file 40 will now be described in greater detail. The Boot Info file 40 may be stored in the System Folder of the start-up volume. Alternatively, the Boot Info file 40 may be provided by a network server using, for example, the Bootstrap Protocol (BootP), which is described in B. Croft et al., Network Working Group Request for Comments (RFC) 951, September 1985. Referring to FIG. 5, the Boot Info file 40 includes Open Firmware-specific Mac OS code 52, referred to as the “Trampoline code”; an Open Firmware header 51, which includes a Forth script that performs operations necessary to the start-up of the OS, including validation tests and transfer of control to the Trampoline code; and a compressed ROM Image 53, which represents the mid-level portion 32 of the OS 30. The purpose of the header 51 is, generally, to specify the locations of the other components of the Boot Info file 40. The purpose of the Trampoline code 53 generally is to handle the transition between the Open Firmware code in the Boot Rom 11 and the ROM Image 52, as will be described in greater detail below.

Esfahani ’695, 7:5-24; *also* Esfahani ’265, 7:9:28.

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image 53 may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored.

Esfahani ’695, 7:43-51; *also* Esfahani ’265, 7:47-55.

> The Boot Info file 40 resides on the boot device (e.g., a disk, or on a network) and has a localizable name.

Esfahani ’695, 7:66-67; *also* Esfahani ’265, 8:3-4.

> Overall Boot Process
>
> Referring now to FIGS. 5, 6A and 6B, the overall boot process of the improved OS 30 will now be described. In response to power on or reset at block 601 (FIG. 6A), the POST code executes (preliminary diagnostics, boot beep, initialization) at 602. At 603, the Open Firmware routine initializes and begins execution, including building the expanded Device Tree and the Interrupt Trees. At 604, the Open Firmware locates the Boot Info file 40, based on defaults and NVRAM settings, and loads the Boot Info file 40 into RAM 12. At 605, the Open Firmware executes the Forth script in the Boot Info file 40, which contains information about the rest of the file (offsets, etc.) and instructions to read both the Trampoline code 52 and the compressed ROM Image 53, and places them into a temporary place in RAM 12. At 606, the Open Firmware transfers control to the Trampoline code 52. At 607, the Trampoline code 52 decompresses the ROM Image 53 in RAM 12. In addition, the Trampoline code reallocates any unused memory space in the ROM Image 52; gathers information about the system from Open Firmware; creates data structures based on this information; terminates Open Firmware, and rearranges the contents of memory to an interim location in physical memory space. At 608, the Trampoline code 52 transfers control to the HardwareInit routine of the (now decompressed) ROM Image 53. At 609, the HardwareInit routine copies data structures to their correct places in memory and then calls the kernel of the OS. Next, at 610 the kernel fills in its data structures and then calls the 68K Emulator. At 611, the 68K Emulator initializes itself and then transfers control to the StartInit routine. At 612, the StartInit routine begins execution, initializing data structures and managers, and booting the MacOS (the high-level portion of the OS). Note that blocks 609 through 612, above, are specific to a Mac OS.

Esfahani ’695, 9:38-10:6; *also* Esfahani ’265, 8:42-9:10.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Esfahani, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> The low-level portion, including hardware-specific code, is stored in a relatively small read-only memory (ROM), while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device, which may be located remotely from the computer system. Upon power-up or reset of the computer system, the code in the ROM is executed to read the compressed ROM image into random access memory (RAM) of the computer system. The compressed image is then decompressed and executed as part of the boot sequence. Once decompressed, the portion of RAM storing the intermediate-level code is write-protected in the memory map, and the code in boot ROM is deleted from the memory map. Memory space in RAM that is allocated to the intermediate-level code but not used is returned to the operating system for use as part of system RAM.

Esfahani '695 [Abstract]

> A method and apparatus for use in booting a computer system are provided. The method includes loading a compressed image of a first portion of the OS of the computer system into a storage device of the

> computer system, which may be RAM, for example. The compressed image of the first portion of the OS is then decompressed and executed as part of the boot sequence of the computer system.

Esfahani '695 2:6-12; *also* Esfahani '265, 2:10-17.

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani 695 2:13-25; *also* Esfahani '265, 2:18-29.

> A computer OS using a compressed ROM image in RAM is described. In brief, the low-level portion of an OS of a computer is designed to be separate from the intermediate-level portion of the OS. The low-level portion, which includes hardware-specific code, is stored in a relatively small Boot ROM, while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device. The mass storage device may be located remotely from the computer system, such as in a file server. Upon power-up or reset of the computer system, the code in the boot ROM is executed to read the compressed ROM image into RAM, i.e., system memory, of the computer system. The compressed image is then decompressed and executed as part of the boot sequence.

Esfahani 695, 2:54-67; *also* Esfahani '265, 2:58-3:4.

> Referring now to FIG. 4, in the improved OS, the low-level (hardware-specific) OS code 31 resides in firmware, in order to handle start-up activities of the computer system. This code fits into one, relatively small ROM, referred to as the Boot ROM 11. Thus, Boot ROM 11 includes all of the hardware-specific code and tables needed to start up the computer as well as to boot the OS and provide common hardware access services the OS might require. Note that the Boot ROM code is not specific to the MacOS or to any other OS. All higher-level software resides elsewhere, as will now be described.

> Prior to start-up, the mid-level portion 32 of OS 30 (which corresponds to part of the ToolBox ROM of earlier Macintosh computers) resides in compressed form in a file 40, referred to as Boot Info file 40.

Esfahani ’695, 5:7-23; *also* Esfahani ’265, 5:10-24.

> During start-up, the Boot Info file 40 is loaded into RAM 12, and the compressed mid-level OS 32 is decompressed. Hence, the mid-level OS 32 is essentially a compressed ROM image. The mid-level OS 32 is inserted into the memory map of the computer system as if it were firmware in ROM. That is, the ROM image can be write-protected in the memory map.

Esfahani ’695, 5:44-51; *also* Esfahani ’265, 5:48-54.

> Referring now to FIG. 5, the low-level OS portion 31 and the Boot Info file 40 are illustrated in greater detail. The low-level portion 31 stored in Boot ROM 11 contains the code needed to start up the computer, initialize and examine the hardware, provide a Device Tree (per Open Firmware) to describe the hardware, provide hardware access services, and transfer control to the OS. In one embodiment, the components of the low-level portion 31 include:
>
> > code for performing Power-On Self Test (POST), including code for performing diagnostics, generating a boot beep and an error beep;
> >
> > Open Firmware code;
> >
> > hardware-specific Mac OS drivers (“ndrv's”) that are needed at boot time (drivers needed at boot time, e.g.,video drivers, network drivers, or disk drivers, are loaded from the Device Tree);
> >
> > HardwareInit code (i.e., the lowest-level code for initializing the CPU, RAM, clock, system bus, etc.) without Mac OS-specific code;
> >
> > code for performing Run-Time Abstraction Services (RTAS). Certain hardware devices differ from machine to machine, but provide similar functions. RTAS provides such functions, including functions for accessing the real-time clock, NVRAM 20, restart, shutdown, and PCI configuration cycles. The I/O

> primitives for these functions in the ROM Image make use of RTAS.

Esfahani ’695, 6:29-54; *also* Esfahani ’265, 6:31-57.

> The Boot Info file 40 will now be described in greater detail. The Boot Info file 40 may be stored in the System Folder of the start-up volume. Alternatively, the Boot Info file 40 may be provided by a network server using, for example, the Bootstrap Protocol (BootP), which is described in B. Croft et al., Network Working Group Request for Comments (RFC) 951, September 1985. Referring to FIG. 5, the Boot Info file 40 includes Open Firmware-specific Mac OS code 52, referred to as the “Trampoline code”; an Open Firmware header 51, which includes a Forth script that performs operations necessary to the start-up of the OS, including validation tests and transfer of control to the Trampoline code; and a compressed ROM Image 53, which represents the mid-level portion 32 of the OS 30. The purpose of the header 51 is, generally, to specify the locations of the other components of the Boot Info file 40. The purpose of the Trampoline code 53 generally is to handle the transition between the Open Firmware code in the Boot Rom 11 and the ROM Image 52, as will be described in greater detail below.

Esfahani ’695, 7:5-24; *also* Esfahani ’265, 7:9:28.

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image 53 may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored.

Esfahani ’695, 7:43-51; *also* Esfahani ’265, 7:47-55.

> The Boot Info file 40 resides on the boot device (e.g., a disk, or on a network) and has a localizable name.

Esfahani ’695, 7:66-67; *also* Esfahani ’265, 8:3-4.

> Overall Boot Process
>
> Referring now to FIGS. 5, 6A and 6B, the overall boot process of the improved OS 30 will now be described. In response to power on or reset

> at block 601 (FIG. 6A), the POST code executes (preliminary diagnostics, boot beep, initialization) at 602. At 603, the Open Firmware routine initializes and begins execution, including building the expanded Device Tree and the Interrupt Trees. At 604, the Open Firmware locates the Boot Info file 40, based on defaults and NVRAM settings, and loads the Boot Info file 40 into RAM 12. At 605, the Open Firmware executes the Forth script in the Boot Info file 40, which contains information about the rest of the file (offsets, etc.) and instructions to read both the Trampoline code 52 and the compressed ROM Image 53, and places them into a temporary place in RAM 12. At 606, the Open Firmware transfers control to the Trampoline code 52. At 607, the Trampoline code 52 decompresses the ROM Image 53 in RAM 12. In addition, the Trampoline code reallocates any unused memory space in the ROM Image 52; gathers information about the system from Open Firmware; creates data structures based on this information; terminates Open Firmware, and rearranges the contents of memory to an interim location in physical memory space. At 608, the Trampoline code 52 transfers control to the HardwareInit routine of the (now decompressed) ROM Image 53. At 609, the HardwareInit routine copies data structures to their correct places in memory and then calls the kernel of the OS. Next, at 610 the kernel fills in its data structures and then calls the 68K Emulator. At 611, the 68K Emulator initializes itself and then transfers control to the StartInit routine. At 612, the StartInit routine begins execution, initializing data structures and managers, and booting the MacOS (the high-level portion of the OS). Note that blocks 609 through 612, above, are specific to a Mac OS.

Esfahani ’695, 9:38-10:6; *also* Esfahani ’265, 8:42-9:10.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Esfahani, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> Refer now to FIG. 2, which illustrates the structure of a traditional Macintosh OS. The OS 22 provides an interface between the hardware 23 of the computer system and the software applications 21. The OS includes the so-called MacOS 22A, which is the high-level portion of the OS 22. In this context, "high-level" refers to the portion of the OS 22 which is least hardware-specific and has the greatest degree of abstraction. Traditionally, this file would reside on disk or other mass storage device and would typically be the last portion of the OS22 to be invoked during the boot process. The (traditional Macintosh) OS22 also includes the so-called ToolBox ROM code 22B. The ToolBox ROM code 22B is firmware that resides in a ROM. The ToolBox ROM code 22B represents the middle- or intermediate-level and low-level portions of the OS22. In this context, "low-level" refers to the portion of the OS which is most hardware-specific and has the smallest degree of abstraction. The middle-level portion has degrees of hardware-dependence and abstraction lower than those of the high-level OS22A and greater than those of the low-level OS.

Esfahani, '695, 4:1-20; Esfahani '265, 4:5-25

See also Esfahani Fig. 2

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Esfahani, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

> FIG. 1 illustrates a computer system 1 in which the present invention may be implemented. Note that while FIG. 1 illustrates the major components of a computer system, it is not intended to represent any particular architecture or manner of interconnecting the component; such details are not germane to the present invention. The computer system of FIG. 1 may be, for example, an Apple Macintosh computer, such as an Apple iMac computer. As shown, the computer system 1 of FIG. 1 includes a microprocessor 10, a read-only memory (ROM) 11, random access memory (RAM) 12, each connected to a bus system 18. The bus system 18 may include one or more buses connected to each other through various bridges, controllers and/or adapters, such as are well-known in the art. For example, the bus system may include a "system bus" that is connected through an adapter to one or more expansion buses, such as a Peripheral Component Interconnect (PCI) bus, or the like. Also coupled to the bus system 18 are a mass storage device 13, a display device 14, a keyboard 15, a pointing device 16, a communication device 17, and non-volatile RAM (NVRAM) 20. A cache memory 19 is coupled to the microprocessor 10.

Esfahani, '695, 3:1-22; Esfahani '265, 3:4-26

> Operation of the Trampoline code is described now in further detail with reference to FIGS. 7A through 7D. At 701, the debugging level for subsequent messages, if any, is determined. More specifically, a node in the Device Tree is checked for a specific property, and if the property exists, then its value is used to determine the debugging level. At 702, it

> is determined whether to create the read/write Debug ROM Image aperture (a specific level of debugging will cause the Debug ROM Image aperture to be created), and at 703 the presence of the appropriate copyright notice in the Boot ROM 11 is verified. At 704, the memory controller and the main memory node are located. At 705, the Trampoline code allocates RAM for the RTAS and instantiates the RTAS. At 706, an appropriate amount of RAM (e.g., 768K in one embodiment) is allocated for a Work Area that will contain the Device Tree and various other data structures. At this point, in certain embodiments, information on the hardware configuration of the system (e.g., processor, memory devices, etc.) may be gathered from the Device Tree and saved in one or more records. At 707, space is allocated in RAM for the ROM Image 53, and the ROM Image 53 is then decompressed from the Boot Info file into the allocated space.

Esfahani, ’695, 9:53-10:8; Esfahani ’265, 9:57-10:11

> At 712, the interrupt controller(s) are initialized, and at 713 the memory map is built. At 714, the ROM Image checksum is verified, and at 715, the physical addresses of the data structures in the Work Area are determined in preparation for moving these data structured to their proper locations. At 716, the Trampoline code causes Open Firmware to quiesce, while allowing Open Firmware to shut down any active hardware that might be performing DMA or interrupts. At 717, the Trampoline code switches from virtual to real mode, moves the ROM Image to its permanent location in RAM, and moves the contents of the Work Area to interim locations. At 718, control is transferred to the HardwareInit routine of the ROM Image. At 719, the Work Area data structures are moved to the their permanent locations, and at 720, control is transferred from the HardwareInit code to the kernel.

Esfahani, ’695, 10:21-35; Esfahani ’265, 10:25-40

See also Esfahani Fig. 1

4. The method of claim 1, further comprising updating the list of boot data.	Esfahani, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claim 1.1 above.

See also

> Referring now to FIG. 4, in the improved OS, the low-level (hardware-specific) OS code 31 resides in firmware, in order to handle start-up activities of the computer system. This code fits into one, relatively small ROM, referred to as the Boot ROM 11. Thus, Boot ROM 11 includes all of the hardware-specific code and tables needed to start up the computer as well as to boot the OS and provide common hardware access services the OS might require. Note that the Boot ROM code is not specific to the MacOS or to any other OS. All higher-level software resides elsewhere, as will now be described.
>
> Prior to start-up, the mid-level portion 32 of OS 30 (which corresponds to part of the ToolBox ROM of earlier Macintosh computers) resides in compressed form in a file 40, referred to as Boot Info file 40.

Esfahani '695, 5:7-23; *also* Esfahani '265, 5:10-24.

> During start-up, the Boot Info file 40 is loaded into RAM 12, and the compressed mid-level OS 32 is decompressed. Hence, the mid-level OS 32 is essentially a compressed ROM image. The mid-level OS 32 is inserted into the memory map of the computer system as if it were firmware in ROM. That is, the ROM image can be write-protected in the memory map.

Esfahani '695, 5:44-51; *also* Esfahani '265, 5:48-54.

> Referring now to FIG. 5, the low-level OS portion 31 and the Boot Info file 40 are illustrated in greater detail. The low-level portion 31 stored in Boot ROM 11 contains the code needed to start up the computer, initialize and examine the hardware, provide a Device Tree (per Open

> Firmware) to describe the hardware, provide hardware access services, and transfer control to the OS. In one embodiment, the components of the low-level portion 31 include:
>
> > code for performing Power-On Self Test (POST), including code for performing diagnostics, generating a boot beep and an error beep;
> >
> > Open Firmware code;
> >
> > hardware-specific Mac OS drivers ("ndrv's") that are needed at boot time (drivers needed at boot time, e.g.,video drivers, network drivers, or disk drivers, are loaded from the Device Tree);
> >
> > HardwareInit code (i.e., the lowest-level code for initializing the CPU, RAM, clock, system bus, etc.) without Mac OS-specific code;
> >
> > code for performing Run-Time Abstraction Services (RTAS). Certain hardware devices differ from machine to machine, but provide similar functions. RTAS provides such functions, including functions for accessing the real-time clock, NVRAM 20, restart, shutdown, and PCI configuration cycles. The I/O primitives for these functions in the ROM Image make use of RTAS.

Esfahani '695, 6:29-54; *also* Esfahani '265, 6:31-57.

> The Boot Info file 40 will now be described in greater detail. The Boot Info file 40 may be stored in the System Folder of the start-up volume. Alternatively, the Boot Info file 40 may be provided by a network server using, for example, the Bootstrap Protocol (BootP), which is described in B. Croft et al., Network Working Group Request for Comments (RFC) 951, September 1985. Referring to FIG. 5, the Boot Info file 40 includes Open Firmware-specific Mac OS code 52, referred to as the "Trampoline code"; an Open Firmware header 51, which includes a Forth script that performs operations necessary to the start-up of the OS, including validation tests and transfer of control to the Trampoline code; and a compressed ROM Image 53, which represents the mid-level portion 32 of the OS 30. The purpose of the header 51 is, generally, to specify the locations of the other components of the Boot Info file 40. The purpose of the Trampoline code 53 generally is to handle the transition between the Open Firmware code in the Boot Rom 11 and the ROM Image 52, as will be described in greater detail below.

Esfahani ’695, 7:5-24; *also* Esfahani ’265, 7:9:28.

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image 53 may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored.

Esfahani ’695, 7:43-51; *also* Esfahani ’265, 7:47-55.

> The Boot Info file 40 resides on the boot device (e.g., a disk, or on a network) and has a localizable name.

Esfahani ’695, 7:66-67; *also* Esfahani ’265, 8:3-4.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Esfahani, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Esfahani, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Esfahani, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Esfahani, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

> FIG. 1 illustrates a computer system 1 in which the present invention may be implemented. Note that while FIG. 1 illustrates the major components of a computer system, it is not intended to represent any particular architecture or manner of interconnecting the component; such details are not germane to the present invention. The computer system of FIG. 1 may be, for example, an Apple Macintosh computer, such as an Apple iMac computer. As shown, the computer system 1 of FIG. 1 includes a microprocessor 10, a read-only memory (ROM) 11, random access memory (RAM) 12, each connected to a bus system 18. The bus system 18 may include one or more buses connected to each other through various bridges, controllers and/or adapters, such as are well-known in the art. For example, the bus system may include a "system bus" that is connected through an adapter to one or more expansion buses, such as a Peripheral Component Interconnect (PCI) bus, or the like. Also coupled to the bus system 18 are a mass storage device 13, a display device 14, a keyboard 15, a pointing device 16, a communication device 17, and non-volatile RAM (NVRAM) 20. A cache memory 19 is coupled to the microprocessor 10.

Esfahani, '695, 3:1-22; Esfahani '265, 3:4-26

> Operation of the Trampoline code is described now in further detail with reference to FIGS. 7A through 7D. At 701, the debugging level for subsequent messages, if any, is determined. More specifically, a node in the Device Tree is checked for a specific property, and if the property exists, then its value is used to determine the debugging level. At 702, it is determined whether to create the read/write Debug ROM Image aperture (a specific level of debugging will cause the Debug ROM

> Image aperture to be created), and at 703 the presence of the appropriate copyright notice in the Boot ROM 11 is verified. At 704, the memory controller and the main memory node are located. At 705, the Trampoline code allocates RAM for the RTAS and instantiates the RTAS. At 706, an appropriate amount of RAM (e.g., 768K in one embodiment) is allocated for a Work Area that will contain the Device Tree and various other data structures. At this point, in certain embodiments, information on the hardware configuration of the system (e.g., processor, memory devices, etc.) may be gathered from the Device Tree and saved in one or more records. At 707, space is allocated in RAM for the ROM Image 53, and the ROM Image 53 is then decompressed from the Boot Info file into the allocated space.

Esfahani, ’695, 9:53-10:8; Esfahani ’265, 9:57-10:11

> At 712, the interrupt controller(s) are initialized, and at 713 the memory map is built. At 714, the ROM Image checksum is verified, and at 715, the physical addresses of the data structures in the Work Area are determined in preparation for moving these data structured to their proper locations. At 716, the Trampoline code causes Open Firmware to quiesce, while allowing Open Firmware to shut down any active hardware that might be performing DMA or interrupts. At 717, the Trampoline code switches from virtual to real mode, moves the ROM Image to its permanent location in RAM, and moves the contents of the Work Area to interim locations. At 718, control is transferred to the HardwareInit routine of the ROM Image. At 719, the Work Area data structures are moved to the their permanent locations, and at 720, control is transferred from the HardwareInit code to the kernel.

Esfahani, ’695, 10:21-35; Esfahani ’265, 10:25-40

See also Esfahani Fig. 1

7.2 a cache memory device; and;	Esfahani, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Esfahani, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Esfahani, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Esfahani, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Esfahani, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> Refer now to FIG. 2, which illustrates the structure of a traditional Macintosh OS. The OS 22 provides an interface between the hardware 23 of the computer system and the software applications 21. The OS includes the so-called MacOS 22A, which is the high-level portion of the OS 22. In this context, "high-level" refers to the portion of the OS 22 which is least hardware-specific and has the greatest degree of abstraction. Traditionally, this file would reside on disk or other mass storage device and would typically be the last portion of the OS22 to be invoked during the boot process. The (traditional Macintosh) OS22 also includes the so-called ToolBox ROM code 22B. The ToolBox ROM code 22B is firmware that resides in a ROM. The ToolBox ROM code 22B represents the middle- or intermediate-level and low-level portions of the OS22. In this context, "low-level" refers to the portion of the OS which is most hardware-specific and has the smallest degree of

abstraction. The middle-level portion has degrees of hardware-dependence and abstraction lower than those of the high-level OS22A and greater than those of the low-level OS.

Esfahani, ’695, 4:1-20; Esfahani ’265, 4:5-25

See also Esfahani Fig. 2

9.1 maintaining a list of application data associated with an application program;	Esfahani, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Esfahani, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Esfahani, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> Refer now to FIG. 2, which illustrates the structure of a traditional Macintosh OS. The OS 22 provides an interface between the hardware 23 of the computer system and the software applications 21. The OS includes the so-called MacOS 22A, which is the high-level portion of the OS 22. In this context, "high-level" refers to the portion of the OS 22 which is least hardware-specific and has the greatest degree of abstraction. Traditionally, this file would reside on disk or other mass storage device and would typically be the last portion of the OS22 to be invoked during the boot process. The (traditional Macintosh) OS22 also includes the so-called ToolBox ROM code 22B. The ToolBox ROM code 22B is firmware that resides in a ROM. The ToolBox ROM code 22B represents the middle- or intermediate-level and low-level portions of the OS22. In this context, "low-level" refers to the portion of the OS which is most hardware-specific and has the smallest degree of abstraction. The middle-level portion has degrees of hardware-

dependence and abstraction lower than those of the high-level OS22A and greater than those of the low-level OS.

Esfahani, ’695, 4:1-20; Esfahani ’265, 4:5-25

See also Esfahani Fig. 2

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Esfahani, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> A computer OS using a compressed ROM image in RAM is described. In brief, the low-level portion of an OS of a computer is designed to be separate from the intermediate-level portion of the OS. The low-level portion, which includes hardware-specific code, is stored in a relatively small Boot ROM, while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device. The mass storage device may be located remotely from the computer system, such as in a file server. Upon power-up or reset of the computer system, the code in the boot ROM is executed to read the compressed ROM image into RAM, i.e., system memory, of the computer system. The compressed image is then decompressed and executed as part of the boot sequence.

Esfahani 695, 2:54-67; *also* Esfahani '265, 2:58-3:4.

> Referring now to FIG. 4, in the improved OS, the low-level (hardware-specific) OS code 31 resides in firmware, in order to handle start-up activities of the computer system. This code fits into one, relatively small ROM, referred to as the Boot ROM 11. Thus, Boot ROM 11 includes all of the hardware-specific code and tables needed to start up the computer as well as to boot the OS and provide common hardware access services the OS might require. Note that the Boot ROM code is not specific to the MacOS or to any other OS. All higher-level software resides elsewhere, as will now be described.

> Prior to start-up, the mid-level portion 32 of OS 30 (which corresponds to part of the ToolBox ROM of earlier Macintosh computers) resides in compressed form in a file 40, referred to as Boot Info file 40.

Esfahani ’695, 5:7-23; *also* Esfahani ’265, 5:10-24.

> During start-up, the Boot Info file 40 is loaded into RAM 12, and the compressed mid-level OS 32 is decompressed. Hence, the mid-level OS 32 is essentially a compressed ROM image. The mid-level OS 32 is inserted into the memory map of the computer system as if it were firmware in ROM. That is, the ROM image can be write-protected in the memory map.

Esfahani ’695, 5:44-51; *also* Esfahani ’265, 5:48-54.

> Referring now to FIG. 5, the low-level OS portion 31 and the Boot Info file 40 are illustrated in greater detail. The low-level portion 31 stored in Boot ROM 11 contains the code needed to start up the computer, initialize and examine the hardware, provide a Device Tree (per Open Firmware) to describe the hardware, provide hardware access services, and transfer control to the OS. In one embodiment, the components of the low-level portion 31 include:
>
> > code for performing Power-On Self Test (POST), including code for performing diagnostics, generating a boot beep and an error beep;
> >
> > Open Firmware code;
> >
> > hardware-specific Mac OS drivers (“ndrv's”) that are needed at boot time (drivers needed at boot time, e.g.,video drivers, network drivers, or disk drivers, are loaded from the Device Tree);
> >
> > HardwareInit code (i.e., the lowest-level code for initializing the CPU, RAM, clock, system bus, etc.) without Mac OS-specific code;
> >
> > code for performing Run-Time Abstraction Services (RTAS). Certain hardware devices differ from machine to machine, but provide similar functions. RTAS provides such functions, including functions for accessing the real-time clock, NVRAM 20, restart, shutdown, and PCI configuration cycles. The I/O primitives for these functions in the ROM Image make use of RTAS.

Esfahani ’695, 6:29-54; *also* Esfahani ’265, 6:31-57.

> The Boot Info file 40 will now be described in greater detail. The Boot Info file 40 may be stored in the System Folder of the start-up volume. Alternatively, the Boot Info file 40 may be provided by a network server using, for example, the Bootstrap Protocol (BootP), which is described in B. Croft et al., Network Working Group Request for Comments (RFC) 951, September 1985. Referring to FIG. 5, the Boot Info file 40 includes Open Firmware-specific Mac OS code 52, referred to as the “Trampoline code”; an Open Firmware header 51, which includes a Forth script that performs operations necessary to the start-up of the OS, including validation tests and transfer of control to the Trampoline code; and a compressed ROM Image 53, which represents the mid-level portion 32 of the OS 30. The purpose of the header 51 is, generally, to specify the locations of the other components of the Boot Info file 40. The purpose of the Trampoline code 53 generally is to handle the transition between the Open Firmware code in the Boot Rom 11 and the ROM Image 52, as will be described in greater detail below.

Esfahani ’695, 7:5-24; *also* Esfahani ’265, 7:9:28.

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image 53 may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored.

Esfahani ’695, 7:43-51; *also* Esfahani ’265, 7:47-55.

> The Boot Info file 40 resides on the boot device (e.g., a disk, or on a network) and has a localizable name.

Esfahani ’695, 7:66-67; *also* Esfahani ’265, 8:3-4.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Esfahani, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> The low-level portion, including hardware-specific code, is stored in a relatively small read-only memory (ROM), while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device, which may be located remotely from the computer system. Upon power-up or reset of the computer system, the code in the ROM is executed to read the compressed ROM image into random access memory (RAM) of the computer system. The compressed image is then decompressed and executed as part of the boot sequence. Once decompressed, the portion of RAM storing the intermediate-level code is write-protected in the memory map, and the code in boot ROM is deleted from the memory map. Memory space in RAM that is allocated to the intermediate-level code but not used is returned to the operating system for use as part of system RAM.

Esfahani '695 [Abstract]

> A method and apparatus for use in booting a computer system are provided. The method includes loading a compressed image of a first portion of the OS of the computer system into a storage device of the computer system, which may be RAM, for example. The compressed image of the first portion of the OS is then decompressed and executed as part of the boot sequence of the computer system.

Esfahani '695 2:6-12; *also* Esfahani '265, 2:10-17.

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the

> portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani 695 2:13-25; *also* Esfahani '265, 2:18-29.

> A computer OS using a compressed ROM image in RAM is described. In brief, the low-level portion of an OS of a computer is designed to be separate from the intermediate-level portion of the OS. The low-level portion, which includes hardware-specific code, is stored in a relatively small Boot ROM, while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device. The mass storage device may be located remotely from the computer system, such as in a file server. Upon power-up or reset of the computer system, the code in the boot ROM is executed to read the compressed ROM image into RAM, i.e., system memory, of the computer system. The compressed image is then decompressed and executed as part of the boot sequence.

Esfahani 695, 2:54-67; *also* Esfahani '265, 2:58-3:4.

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image 53 may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored.

Esfahani '695, 7:43-51; *also* Esfahani '265, 7:47-55.

> Overall Boot Process
>
> Referring now to FIGS. 5, 6A and 6B, the overall boot process of the improved OS 30 will now be described. In response to power on or reset at block 601 (FIG. 6A), the POST code executes (preliminary diagnostics, boot beep, initialization) at 602. At 603, the Open Firmware routine initializes and begins execution, including building the expanded Device Tree and the Interrupt Trees. At 604, the Open Firmware locates the Boot Info file 40, based on defaults and NVRAM settings, and loads the Boot Info file 40 into RAM 12. At 605, the Open Firmware executes the Forth script in the Boot Info file 40, which contains information about the rest of the file (offsets, etc.) and instructions to read both the Trampoline code 52 and the compressed

ROM Image 53, and places them into a temporary place in RAM 12. At 606, the Open Firmware transfers control to the Trampoline code 52. At 607, the Trampoline code 52 decompresses the ROM Image 53 in RAM 12. In addition, the Trampoline code reallocates any unused memory space in the ROM Image 52; gathers information about the system from Open Firmware; creates data structures based on this information; terminates Open Firmware, and rearranges the contents of memory to an interim location in physical memory space. At 608, the Trampoline code 52 transfers control to the HardwareInit routine of the (now decompressed) ROM Image 53. At 609, the HardwareInit routine copies data structures to their correct places in memory and then calls the kernel of the OS. Next, at 610 the kernel fills in its data structures and then calls the 68K Emulator. At 611, the 68K Emulator initializes itself and then transfers control to the StartInit routine. At 612, the StartInit routine begins execution, initializing data structures and managers, and booting the MacOS (the high-level portion of the OS). Note that blocks 609 through 612, above, are specific to a Mac OS.

Esfahani ’695, 9:38-10:6; *also* Esfahani ’265, 8:42-9:10.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Esfahani, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Esfahani, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.3 and 1.4 above.

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani, '695, 2:14-25; Esfahani '265, 2:17-29

> FIG. 1 illustrates a computer system 1 in which the present invention may be implemented. Note that while FIG. 1 illustrates the major components of a computer system, it is not intended to represent any particular architecture or manner of interconnecting the component; such details are not germane to the present invention. The computer system of FIG. 1 may be, for example, an Apple Macintosh computer, such as an Apple iMac computer. As shown, the computer system 1 of FIG. 1 includes a microprocessor 10, a read-only memory (ROM) 11, random access memory (RAM) 12, each connected to a bus system 18. The bus system 18 may include one or more buses connected to each other through various bridges, controllers and/or adapters, such as are well-known in the art. For example, the bus system may include a "system bus" that is connected through an adapter to one or more expansion buses, such as a Peripheral Component Interconnect (PCI) bus, or the like. Also coupled to the bus system 18 are a mass storage device 13, a display device 14, a keyboard 15, a pointing device 16, a

> communication device 17, and non-volatile RAM (NVRAM) 20. A cache memory 19 is coupled to the microprocessor 10.

Esfahani, ’695, 3:1-22; Esfahani ’265, 3:4-26

> The Boot Info file 40 resides on the boot device (e.g., a disk, or on a network) and has a localizable name. Identification information that leads to the file's path may be stored in NVRAM 20 or another suitable location, and the search algorithm for a usable Boot Info file parallels the search mechanism across SCSI, ATA, etc., used in the earlier Macintosh OS's StartSearch routine. By default, the Boot Info file 40 is located by using the current, active System folder's dirID (directory ID) in the boot block of each partition of the Hierarchical File System (HFS) and then searching for a file with a predetermined file type. Searching by file type is done to allow localization of the file.

Esfahani, ’695, 7:66-8:10; Esfahani ’265, 8:3-14

See also Esfahani Fig. 1

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Esfahani, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.3 and 1.4 above.

See also

> The ROM Image 53 of the improved OS 30 is similar to the ToolBox ROM code of the earlier Macintosh OS, in that it has a similar layout and contains many of the same components. The image may be compressed using a known compression algorithm, such as LZSS, for example. The ROM Image 53 may also be encoded, if desired.

Esfahani, '695, 8:32-37; Esfahani '265, 8:36-41

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Esfahani, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

See also

> The ROM Image 53 of the improved OS 30 is similar to the ToolBox ROM code of the earlier Macintosh OS, in that it has a similar layout and contains many of the same components. The image may be compressed using a known compression algorithm, such as LZSS, for example. The ROM Image 53 may also be encoded, if desired.

Esfahani, '695, 8:32-37; Esfahani '265, 8:36-41

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Esfahani, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

Appendix A1
Invalidity of U.S. Patent 7,181,608 based on Esfahani

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Esfahani, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Esfahani, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Esfahani, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Esfahani, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Esfahani, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Esfahani, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Esfahani, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Esfahani, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Esfahani, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Esfahani, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Esfahani, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Esfahani, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Esfahani, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Esfahani, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Esfahani, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Esfahani, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Esfahani, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 15 and 27 above	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Esfahani, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 16 and 27 above	

Appendix A2
Invalidity of U.S. Patent 7,181,608 based on Lillich

U.S. Patent Nos. 5,619,698 to Lillich ("Lillich '698) and 5,790,856 to Lillich ("Lillich '856"), alone or in combination, invalidate claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

In addition, the prior art 68k operating system disclosed in Lillich (see Lillich '856, 1:20-5:55, and Lillich '698, 1:10-5:30) ("68k System") laims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of the '608 Patent pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 because the system was on sale or in public use more than one year prior to the filing date of the '608 patent. Lillich '698, Lillich '856, and the 68k System are collectively referred to herein as "Lillich."

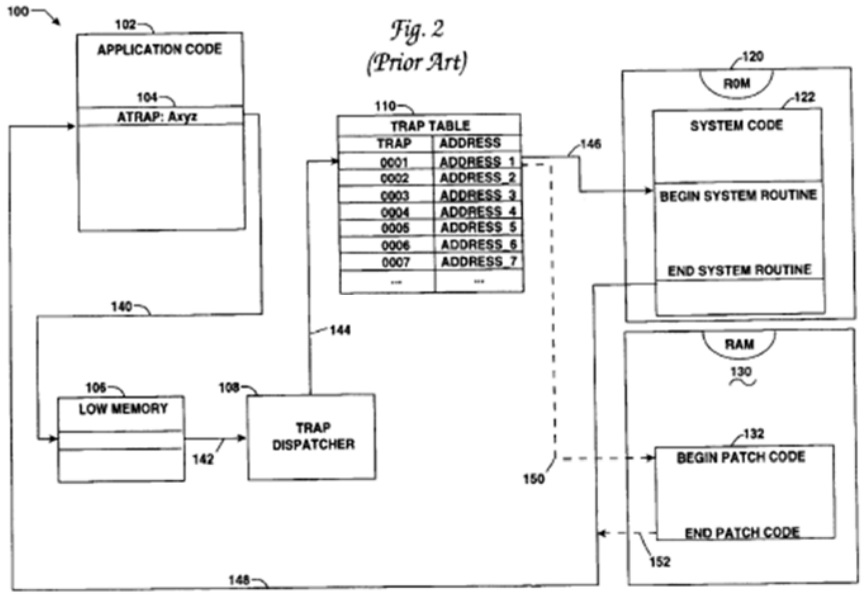
Additionally, Apple intends to provide and seek discovery related to this system to obtain copies of relevant materials, including but not limited to, architectural documents, design documents, implementation documents, internal publications, patent applications and business records relating to this product and will supplement these contentions after those materials are discovered or received. Apple also reserves its rights to rely on any additional 68k System materials, either individually or collectively, as prior art publications to the '608 patent.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Lillich, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this claim limitation:



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as

other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

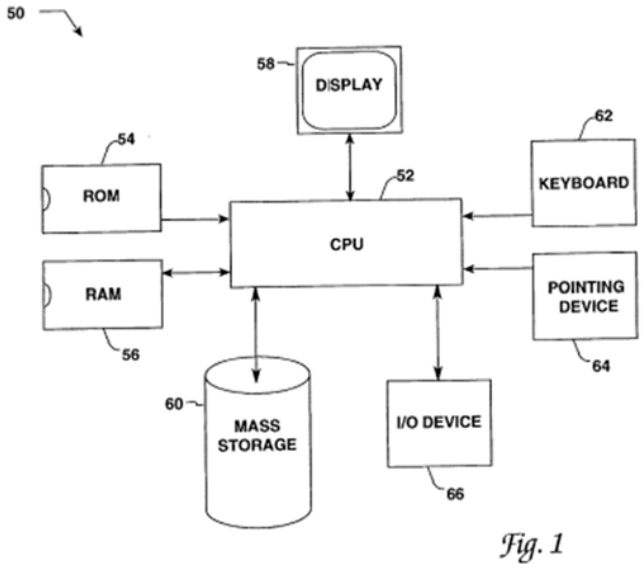


Fig. 1

Lillich, Fig. 1.

"By way of example, a representative computer system 50 is illustrated schematically in FIG. 1."

Lillich '856, 8:44-46; also Lillich '698, 9:39-41.

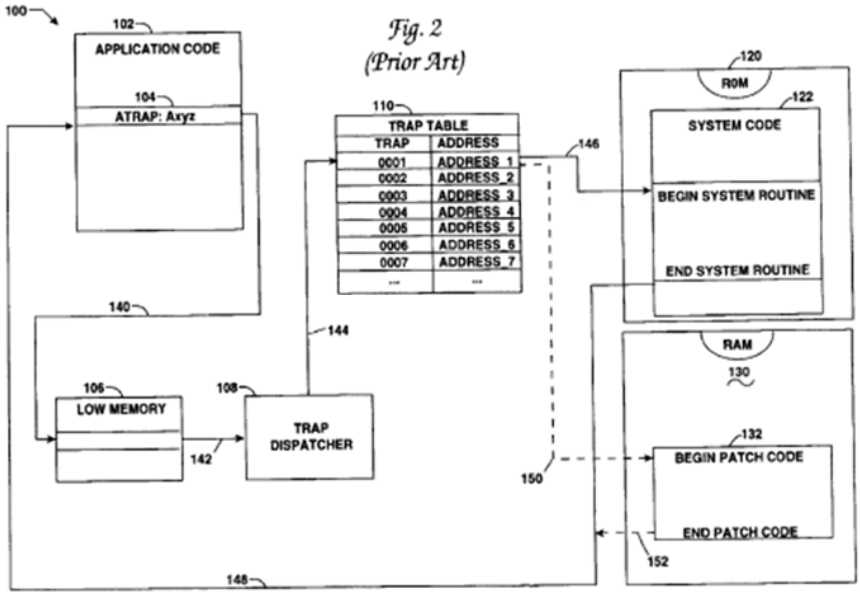
"As will be appreciated by those skilled in the art. CPU 52 includes a microprocessor and any additional circuitry and/ or device drivers necessary to control the computer system. For instance. the CPU 52 may include a keyboard controller which provides an interface between the microprocessor and the keyboard 62. ROM 64 is typically persistent memory accessible by the CPU 52 which contains the operating system instructions either in an executable format or in a compressed format which is expanded when the computer system 50 boots. RAM 56 is typically transient memory and is used as "scratch pad" memory by the operating system and/or any applications implemented on the computer system 50. For example. if a portion of the operating system present in ROM 64 is in compressed format, it may be expanded and stored into RAM 56."

Lillich '856, 8:53-67; *also* Lillich '698, 9:49-63.

1.1 maintaining a list of boot data used for booting a computer system;	Lillich, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this claim limitation:



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as

> other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

> "The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

> "While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

1.2 initializing a central processing unit of the computer system;	Lillich, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this claim limitation:

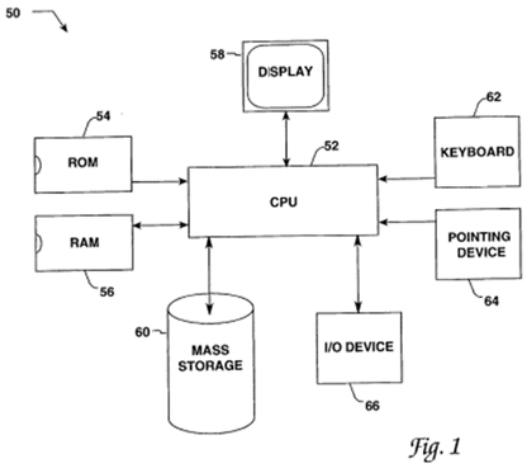


Fig. 1

Lillich, Fig. 1.

> "By way of example, a representative computer system 50 is illustrated schematically in FIG. 1."

Lillich '856, 8:44-46; *also* Lillich '698, 9:39-41.

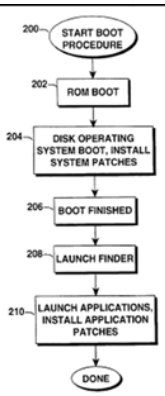


Fig. 3
(Prior Art)

Lillich '698, Fig 3.

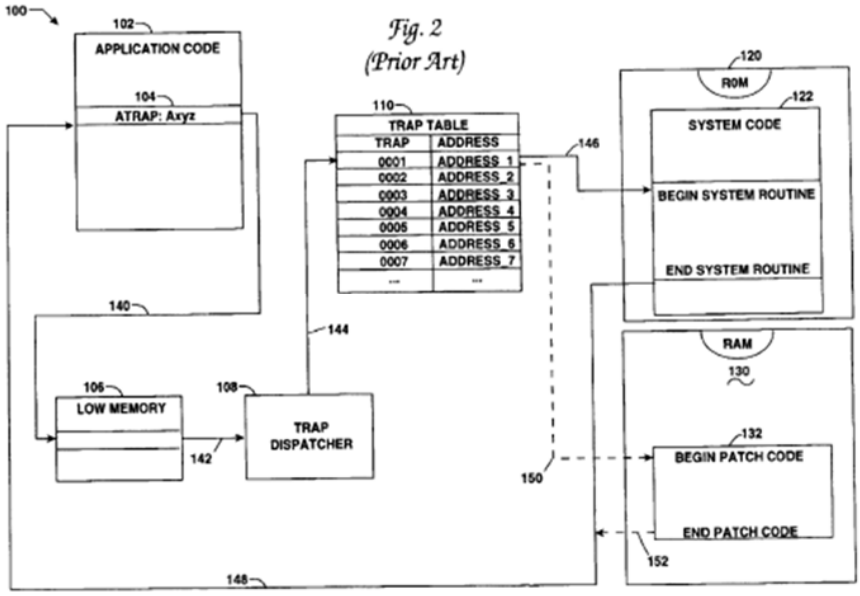
> "Next, in reference to FIG. 3, one method for installing patches in the 68K operating system will be described. In an initial step 200, the computer boot process is started. Then, in a step 202, the ROM boot is performed. In ROM boot step 202, initialization procedures such as expanding the trap table from ROM into RAM are performed. Next, in a step 204, the disk operating system boot is performed. At this point any system patches are also installed."

Lillich '698, 3:52-59.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Lillich, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this claim limitation:



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which

> operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

> "The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

> "While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

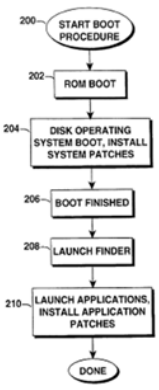


Fig. 3
(Prior Art)

Lillich '698, Fig 3.

> "Next, in reference to FIG. 3, one method for installing patches in the

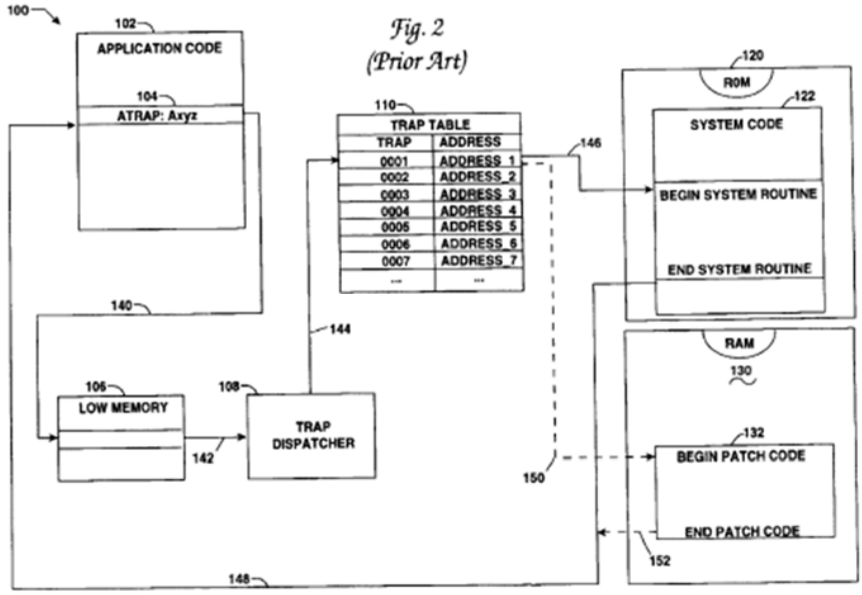
> 68K operating system will be described. In an initial step 200, the computer boot process is started. Then, in a step 202, the ROM boot is performed. In ROM boot step 202, initialization procedures such as expanding the trap table from ROM into RAM are performed. Next, in a step 204, the disk operating system boot is performed. At this point any system patches are also installed."

Lillich '698, 3:52-59.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Lillich, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this claim limitation:



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

> "The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

> "While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

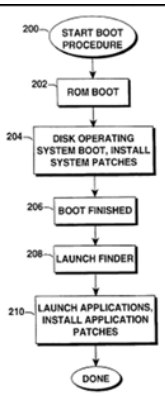


Fig. 3
(Prior Art)

Lillich ’698, Fig 3.

> “Next, in reference to FIG. 3, one method for installing patches in the 68K operating system will be described. In an initial step 200, the computer boot process is started. Then, in a step 202, the ROM boot is performed. In ROM boot step 202, initialization procedures such as expanding the trap table from ROM into RAM are performed. Next, in a step 204, the disk operating system boot is performed. At this point any system patches are also installed.”

Lillich ’698, 3:52-59.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Lillich, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> The system routines for the 68K operating system reside mainly in ROM. However, to provide flexibility for any subsequent development, application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM. This indirect mechanism permits the ROM addressing of system routines to vary, or to be replaced by patch routines, without affecting the operation of applications which utilize the system routines.

Lillich '856, 1:52-61; *also* Lillich '698, 2:14-23

> Additionally, when new applications are launched they too can load new versions of individual routines into RAM and then patch the trap table in order to redirect any calls to the original system routine to the new versions.

Lillich '856, 2:60-64; *also* Lillich '698, 3:34-37

> In the world of Macintosh®, DLLs can be loosely divided into three different categories: applications, import libraries, and extensions. Typically, applications are fragments which have a user interface and are designed to function interactively with the user. Often applications do not export symbols to other fragments but rather serve as a root

> fragments, providing some root functionality and importing other necessary functionality.
>
> . . .
>
> Once an application has been launched and the binding manager has completed preparation, the end result is a new, self-sufficient executable process with an associated data closure. The closure contains a root fragment as well as instances of all the import libraries required to resolve all the import symbols present in both the root fragment and the import libraries. In explanation, the root fragment is the fragment which includes some root functionality as well as a list of import symbols.

Lillich ’856, 3:40-4:9

> As discussed previously, new versions of individual system routines may be loaded into RAM and the trap table patched in order to redirect any calls to the original system routine to the new versions. After this is complete, the operating system boot procedure is complete in a step 206. In a next step 208, the application "Finder" is launched. As will be appreciated by those skilled in the art, the Finder is the primal application which performs critical tasks such as displaying the Macintosh® desktop and launching other applications at the request of the user. Once the Finder is running, in a step 210 other applications may be launched and in turn any patches necessary for the other applications may be installed.

Lillich ’698, 3:60-4:5

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Lillich, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

> As will be appreciated by those skilled in the art, CPU 52 includes a microprocessor and any additional circuitry and/or device drivers necessary to control the computer system. For instance, the CPU 52 may include a keyboard controller which provides an interface between the microprocessor and the keyboard 62. ROM 64 is typically persistent memory accessible by the CPU 52 which contains the operating system instructions either in an executable format or in a compressed format which is expanded when the computer system 50 boots. RAM 56 is typically transient memory and is used as "scratch pad" memory by the operating system and/or any applications implemented on the computer system 50. For example, if a portion of the operating system present in ROM 64 is in compressed format, it may be expanded and stored into RAM 56.

Lillich '856, 8:53-67; *also* Lillich '698, 9:48-63

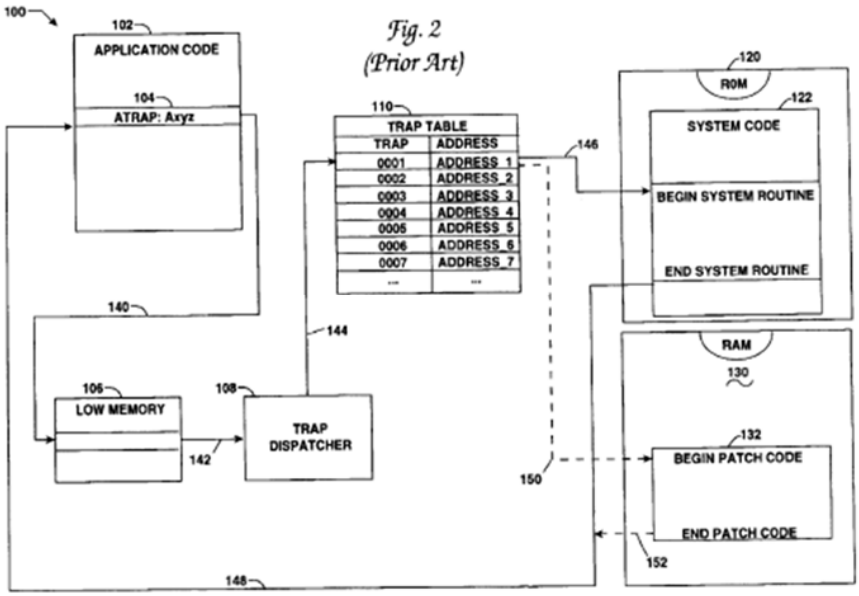
4. The method of claim 1, further comprising updating the list of boot data.	Lillich, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:

See Claim 1.1 above.

See also



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as

other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

"The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

"While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

"Because the trap table 110 is resident in RAM 130. individual entries in the trap table 110 can be changed to point to addresses other than the original ROM addresses. This allows the system routines to be replaced or augmented by patches. At startup time the system can load new versions of individual routines (e.g. from the System file or from a floppy disk) into RAM and then patch the trap table in order to redirect any calls to the original system routine to the new versions."

Lillich '856, 2:52-60, *also* Lillich '698, 3:26-34.

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Lillich, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Lillich, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.1 and 4 above.	

Appendix A2
Invalidity of U.S. Patent 7,181,608 based on Lillich

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Lillich, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Lillich, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

> As will be appreciated by those skilled in the art, CPU 52 includes a microprocessor and any additional circuitry and/or device drivers necessary to control the computer system. For instance, the CPU 52 may include a keyboard controller which provides an interface between the microprocessor and the keyboard 62. ROM 64 is typically persistent memory accessible by the CPU 52 which contains the operating system instructions either in an executable format or in a compressed format which is expanded when the computer system 50 boots. RAM 56 is typically transient memory and is used as "scratch pad" memory by the operating system and/or any applications implemented on the computer system 50. For example, if a portion of the operating system present in ROM 64 is in compressed format, it may be expanded and stored into RAM 56.

Lillich '856, 8:53-67; *also* Lillich '698, 9:48-63

7.2 a cache memory device; and;	Lillich, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Lillich, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Lillich, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Lillich, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Lillich, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> The system routines for the 68K operating system reside mainly in ROM. However, to provide flexibility for any subsequent development, application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM. This indirect mechanism permits the ROM addressing of system routines to vary, or to be replaced by patch routines, without affecting the operation of applications which utilize the system routines.

Lillich '856, 1:52-61; *also* Lillich '698, 2:14-23

> Additionally, when new applications are launched they too can load new versions of individual routines into RAM and then patch the trap

table in order to redirect any calls to the original system routine to the new versions.

Lillich ’856, 2:60-64; *also* Lillich ’698, 3:34-37

In the world of Macintosh®, DLLs can be loosely divided into three different categories: applications, import libraries, and extensions. Typically, applications are fragments which have a user interface and are designed to function interactively with the user. Often applications do not export symbols to other fragments but rather serve as a root fragments, providing some root functionality and importing other necessary functionality.

. . .

Once an application has been launched and the binding manager has completed preparation, the end result is a new, self-sufficient executable process with an associated data closure. The closure contains a root fragment as well as instances of all the import libraries required to resolve all the import symbols present in both the root fragment and the import libraries. In explanation, the root fragment is the fragment which includes some root functionality as well as a list of import symbols.

Lillich ’856, 3:40-4:9

As discussed previously, new versions of individual system routines may be loaded into RAM and the trap table patched in order to redirect any calls to the original system routine to the new versions. After this is complete, the operating system boot procedure is complete in a step 206. In a next step 208, the application "Finder" is launched. As will be appreciated by those skilled in the art, the Finder is the primal application which performs critical tasks such as displaying the Macintosh® desktop and launching other applications at the request of the user. Once the Finder is running, in a step 210 other applications may be launched and in turn any patches necessary for the other applications may be installed.

Lillich ’698, 3:60-4:5

9.1 maintaining a list of application data associated with an application program;	Lillich, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Lillich, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Lillich, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> The system routines for the 68K operating system reside mainly in ROM. However, to provide flexibility for any subsequent development, application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM. This indirect mechanism permits the ROM addressing of system routines to vary, or to be replaced by patch routines, without affecting the operation of applications which utilize the system routines.

Lillich '856, 1:52-61; *also* Lillich '698, 2:14-23

> Additionally, when new applications are launched they too can load new versions of individual routines into RAM and then patch the trap table in order to redirect any calls to the original system routine to the new versions.

Lillich ’856, 2:60-64; *also* Lillich ’698, 3:34-37

> In the world of Macintosh®, DLLs can be loosely divided into three different categories: applications, import libraries, and extensions. Typically, applications are fragments which have a user interface and are designed to function interactively with the user. Often applications do not export symbols to other fragments but rather serve as a root fragments, providing some root functionality and importing other necessary functionality.
>
> . . .
>
> Once an application has been launched and the binding manager has completed preparation, the end result is a new, self-sufficient executable process with an associated data closure. The closure contains a root fragment as well as instances of all the import libraries required to resolve all the import symbols present in both the root fragment and the import libraries. In explanation, the root fragment is the fragment which includes some root functionality as well as a list of import symbols.

Lillich ’856, 3:40-4:9

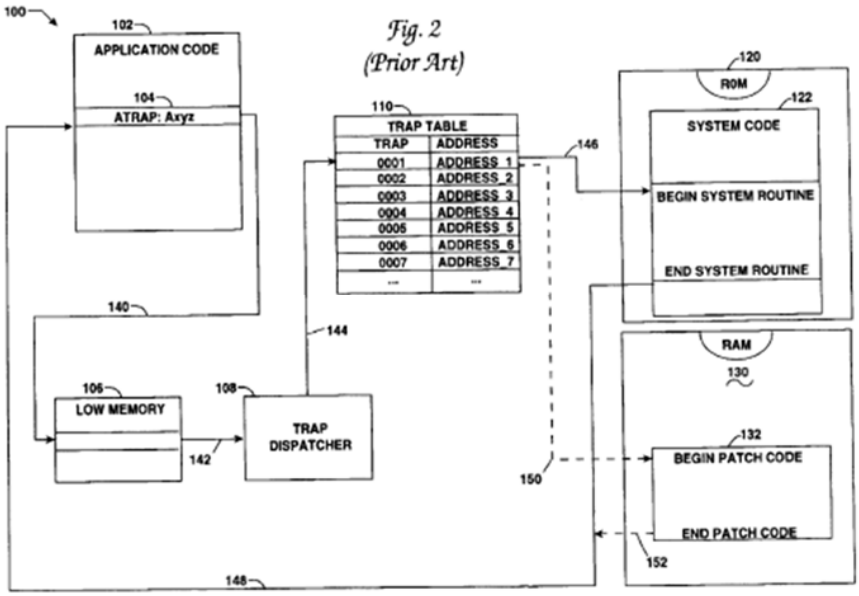
> As discussed previously, new versions of individual system routines may be loaded into RAM and the trap table patched in order to redirect any calls to the original system routine to the new versions. After this is complete, the operating system boot procedure is complete in a step 206. In a next step 208, the application "Finder" is launched. As will be appreciated by those skilled in the art, the Finder is the primal application which performs critical tasks such as displaying the Macintosh® desktop and launching other applications at the request of the user. Once the Finder is running, in a step 210 other applications may be launched and in turn any patches necessary for the other applications may be installed.

Lillich ’698, 3:60-4:5

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Lillich, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this claim limitation:



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such

> as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

> "The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

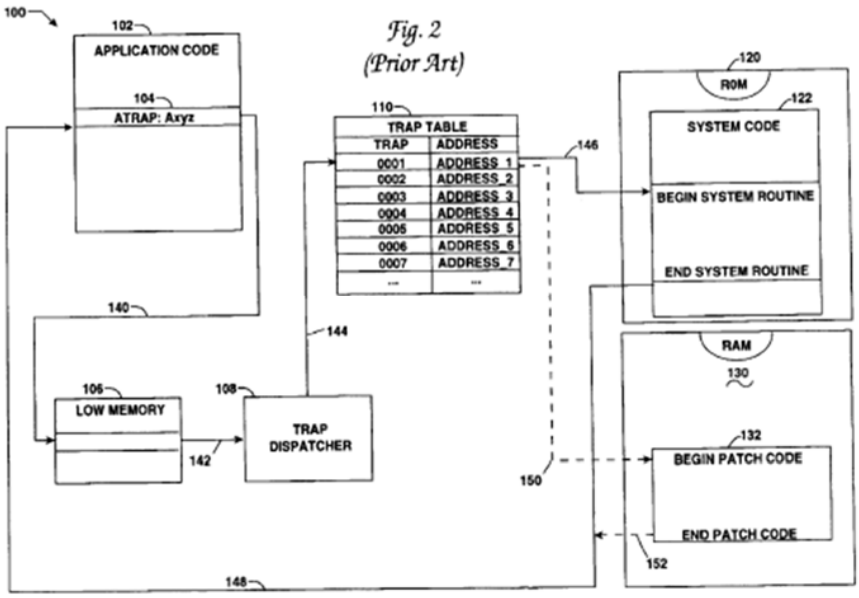
> "While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Lillich, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this claim limitation:



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

> "The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

> "While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Lillich, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Lillich, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Lillich, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Lillich, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.3, 1.4, and 15 above.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Lillich, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Lillich, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Lillich, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Lillich, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Lillich, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Lillich, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Lillich, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Lillich, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Lillich, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Lillich, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Lillich, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Lillich, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Lillich, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Lillich, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Lillich, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Lillich, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Lillich, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Lillich, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 15 and 27 above	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Lillich, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 16 and 27 above	

Appendix A3
Invalidity of U.S. Patent 7,181,608 based on Ballard

U.S. Patent No. 5,933,630 to Ballard ("Ballard") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Ballard, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

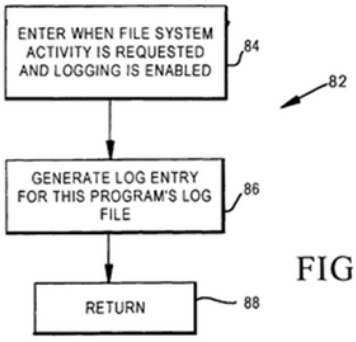


FIG. 4

Ballard, Fig. 4

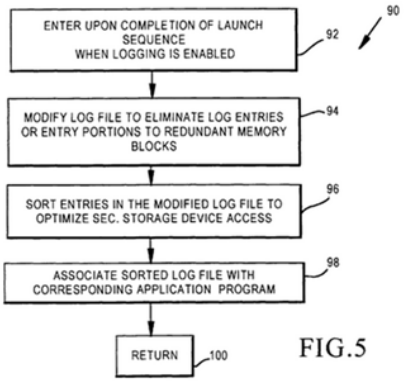


FIG.5

Ballard, Fig. 5

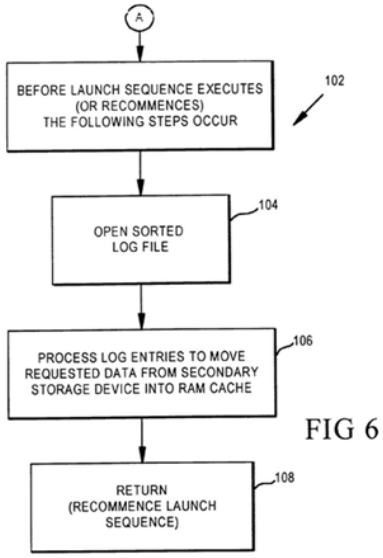


FIG 6

Ballard, Fig. 6

> The processor 12 serves to execute an operating system and one or more application computer programs. In some embodiments there are multiple processors for executing the operating system and application programs. System utilities and/or operating system extension programs also are executed according to some computer system 10 embodiments. Conventional operating systems include DOS, Windows, Windows NT, MacOS, OS/2 and various UNIX-based operating systems. The display device 18 and input devices 20 enable interaction between a user and the computer system 10. The computer system 10 in the process of executing the operating system and zero or more computer programs defines an operating environment for a user to interact with the computer, operating system and executing computer program. The display device 18 serves as

> an output device. Exemplary display devices include a CRT monitor or flat panel display. The user inputs commands and data to the computer system 10 via the input devices. Exemplary input devices include a keyboard, a pointing device and a clicking device. Data also is input to the computer via transportable disks or through I/O ports (not shown).
>
> Ballard, 3:10-30.

1.1 maintaining a list of boot data used for booting a computer system;	Ballard, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> The log file corresponds to a specific computer program--the one being launched that triggered such log file to be created. When multiple programs are being launched at the same time, a log file is created for each such program. The interrupt service routine then sets a flag at step 78 to indicate that logging is enabled for such program. At step 80 the program returns. If the trigger for calling the routine 70 was for a secondary storage device access, then the steps at FIG. 4 also are performed before returning.
>
> Log File System Activity
>
> Once the launch of a computer program is detected and a log file is opened, all file system activity is monitored. Specifically, for each operating system call to the file system the call is analyzed to determine to which application being launched, if any, does the call pertain. Exemplary operating system calls to the file system are OPEN, READ, WRITE, and CLOSE. Referring to FIG. 4 routine 82 is entered at step 84 when both file system activity is detected and logging is enabled. If the call pertains to a computer program being launched, then an entry is appended to the appropriate log file (step 86). The routine 82 then returns at step 88.
>
> The log entry includes a file identifier, the logical address(es) specified in the call and the time of access (e.g., system time; index value). Alternatively, the physical memory address(es) corresponding to the logical address(es) are stored in the log entry. In some embodiments, the operating system has already caused the physical addresses to be generated. If not, then the physical addresses are derived from the logical addresses using the operating system's file allocation table to translate the logical address into the physical address.

A logical address (also referred to as a virtual address) is the address which the computer program uses to access memory. A memory management unit translates this address into a physical address before the actual memory is read or written. A physical address is a memory location on the secondary storage device 16.

Ballard, 5:1-37

Detect Launch Completion

When completion of a launch sequence occurs and logging is enabled, then routine 90 is executed. Referring to FIG. 5, the routine enters at step 92. Conventional operating systems have a specific function that is called when a program is ready for normal execution. Under the Macintosh operating system, the function "Get Next Event" is called. For a computer program running under such operating system, such function is called by the computer program when the launch sequence is complete. According to one embodiment of this invention, step 92 is implemented by an interrupt service routine which is called whenever a computer program calls such function. The interrupt service routine clears the logging enabled flag for the corresponding application program.

In an alternative embodiment, access activity to the secondary storage device 16 is monitored to determine when activity has ceased for a threshold length of time (e.g., 3 seconds). Alternatively or in addition activity is monitored to determine when activity has gone below a threshold data throughput rate (e.g., 50 kilobytes per second) for a threshold period of time (e.g., 5 seconds). When there is insufficient activity for such threshold time, then the program launch is considered to be complete. The routine 90 then is executed.

Modify and Sort Log File

Once the launch sequence is determined to have been completed, then the log entry order is re-organized. The purpose is to eliminate redundant accesses to the same memory block and to optimize access time for the secondary storage device. If accesses occur faster, then the launch time (i.e., time elapsed from start to finish of launch sequence is less) is reduced. Thus, the launch of the computer program is accelerated.

Referring to FIG. 5, at step 94 the log file is processed to eliminate log

entries or log entry portions to redundant memory blocks.

Ballard, 5:55-6:23

> Note in this example that the log includes redundant entries. Entry h2 is a subset of entry d. Entry i2 is the same as entry f. Entry l2 is the same as entry a. It is expected that a redundant access request results from a computer program launch sequence access specifying a different address in the same block as previously accessed.
>
> Frequently the log file will include an entry specifying a single address. Even though only one address is specified, an entire memory block will be accessed (read or written). This is because the minimum file allocation unit is a memory block cluster. Thus, the smallest portion of the computer program than can be accessed is the cluster size (i.e., cluster size). Entries in the log file are tested at step 94 to identify any redundant accesses to portions of the computer program. To determine whether a later entry is an access to a redundant portion of the computer program, the cluster address for the access is identified. The cluster address is either stored in the log file or is derived from the address stored in the log file. For a FAT drive the cluster size is stored in the drive's boot sector. The boot sector includes information about the layout of the file system used for the drive partition. Following the boot sector are several reserved sectors. Following the reserved sectors is the file allocation table (FAT). Following the FAT are one or more backup copies of the FAT. Following the backup copies is the root directory. The size of the root directory is specified in the boot sector. After the root directory are the user's file and directory area. This is the area divided into clusters. Thus, from the information in the boot sector the starting address of cluster space is determined, along with the size of a cluster. Thus, the address boundaries for each cluster are known.
>
> The address for any given cluster x is given by Ax+B, where A and B are constants determined from the boot partition. Thus, for any given file system call the cluster boundaries for such address are determinable. The log file stores either the accessed address, the cluster number (i.e., x) or the cluster address (e.g., start address, end address or some other identifying address) for each cluster accessed.

Ballard, 6:44-7:14

> Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located

> at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.
>
> In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

Ballard, 7:15-53.

See also Ballard, Abstract, Tables B-C, Figs. 4-6.

1.2 initializing a central processing unit of the computer system;	Ballard, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> The log file corresponds to a specific computer program--the one being launched that triggered such log file to be created. When multiple programs are being launched at the same time, a log file is created for each such program. The interrupt service routine then sets a flag at step 78 to indicate that logging is enabled for such program. At step 80 the program returns. If the trigger for calling the routine 70 was for a secondary storage device access, then the steps at FIG. 4 also are performed before returning.
>
> Log File System Activity
>
> Once the launch of a computer program is detected and a log file is opened, all file system activity is monitored. Specifically, for each operating system call to the file system the call is analyzed to determine to which application being launched, if any, does the call pertain. Exemplary operating system calls to the file system are OPEN, READ, WRITE, and CLOSE. Referring to FIG. 4 routine 82 is entered at step 84 when both file system activity is detected and logging is enabled. If the call pertains to a computer program being launched, then an entry is appended to the appropriate log file (step 86). The routine 82 then returns at step 88.
>
> The log entry includes a file identifier, the logical address(es) specified in the call and the time of access (e.g., system time; index value). Alternatively, the physical memory address(es) corresponding to the logical address(es) are stored in the log entry. In some embodiments, the operating system has already caused the physical addresses to be generated. If not, then the physical addresses are derived from the logical addresses using the operating system's file allocation table to translate the logical address into the physical address.

A logical address (also referred to as a virtual address) is the address which the computer program uses to access memory. A memory management unit translates this address into a physical address before the actual memory is read or written. A physical address is a memory location on the secondary storage device 16.

Ballard, 5:1-37

Detect Launch Completion

When completion of a launch sequence occurs and logging is enabled, then routine 90 is executed. Referring to FIG. 5, the routine enters at step 92. Conventional operating systems have a specific function that is called when a program is ready for normal execution. Under the Macintosh operating system, the function "Get Next Event" is called. For a computer program running under such operating system, such function is called by the computer program when the launch sequence is complete. According to one embodiment of this invention, step 92 is implemented by an interrupt service routine which is called whenever a computer program calls such function. The interrupt service routine clears the logging enabled flag for the corresponding application program.

In an alternative embodiment, access activity to the secondary storage device 16 is monitored to determine when activity has ceased for a threshold length of time (e.g., 3 seconds). Alternatively or in addition activity is monitored to determine when activity has gone below a threshold data throughput rate (e.g., 50 kilobytes per second) for a threshold period of time (e.g., 5 seconds). When there is insufficient activity for such threshold time, then the program launch is considered to be complete. The routine 90 then is executed.

Modify and Sort Log File

Once the launch sequence is determined to have been completed, then the log entry order is re-organized. The purpose is to eliminate redundant accesses to the same memory block and to optimize access time for the secondary storage device. If accesses occur faster, then the launch time (i.e., time elapsed from start to finish of launch sequence is less) is reduced. Thus, the launch of the computer program is accelerated.

Referring to FIG. 5, at step 94 the log file is processed to eliminate log

entries or log entry portions to redundant memory blocks.

Ballard, 5:55-6:23

> Note in this example that the log includes redundant entries. Entry h2 is a subset of entry d. Entry i2 is the same as entry f. Entry l2 is the same as entry a. It is expected that a redundant access request results from a computer program launch sequence access specifying a different address in the same block as previously accessed.
>
> Frequently the log file will include an entry specifying a single address. Even though only one address is specified, an entire memory block will be accessed (read or written). This is because the minimum file allocation unit is a memory block cluster. Thus, the smallest portion of the computer program than can be accessed is the cluster size (i.e., cluster size). Entries in the log file are tested at step 94 to identify any redundant accesses to portions of the computer program. To determine whether a later entry is an access to a redundant portion of the computer program, the cluster address for the access is identified. The cluster address is either stored in the log file or is derived from the address stored in the log file. For a FAT drive the cluster size is stored in the drive's boot sector. The boot sector includes information about the layout of the file system used for the drive partition. Following the boot sector are several reserved sectors. Following the reserved sectors is the file allocation table (FAT). Following the FAT are one or more backup copies of the FAT. Following the backup copies is the root directory. The size of the root directory is specified in the boot sector. After the root directory are the user's file and directory area. This is the area divided into clusters. Thus, from the information in the boot sector the starting address of cluster space is determined, along with the size of a cluster. Thus, the address boundaries for each cluster are known.
>
> The address for any given cluster x is given by Ax+B, where A and B are constants determined from the boot partition. Thus, for any given file system call the cluster boundaries for such address are determinable. The log file stores either the accessed address, the cluster number (i.e., x) or the cluster address (e.g., start address, end address or some other identifying address) for each cluster accessed.

Ballard, 6:44-7:14

> Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located

> at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.
>
> In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

Ballard, 7:15-53.

See also Ballard, Abstract, Tables B-C, Figs. 4-6.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Ballard, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> It is common for an application program for a personal computer to be stored in multiple files on the secondary storage device. There often is an executable file, a preferences file and many other files. Some programs include a data base file or a default data file. During a launch of the program multiple files are opened and select portions are moved from the secondary storage device into the primary storage device. When purchasing a computer program the packaging often specifies the amount of RAM address space (i.e., primary storage device address space) required to be allocated to the program while active. By active it is meant that the program has been launched and is currently processing or is

currently able to accept input commands.

Ballard, 1:51-63

> According to another aspect of the invention, the launch access log is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.
>
> A launch sequence as used herein means the sequence of steps executed by the computer system during the launch time which pertain to starting up a given computer program and getting the computer ready to accept input commands for the computer program. A launch sequence is executed for a computer program. Different computer programs have different launch sequences. Steps included in a launch sequence include copying portions of the computer program being launched from the secondary storage device to the primary storage device. Other steps may include allocating a port or device to serve as an input source and/or output receptor. The method of this invention for accelerating a program launch is directed to improving the speed for accessing the secondary storage device during a launch sequence.

Ballard, 3:40-67

> Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located at 100-109, 110-119, 120-129, 130-139, 140-149. When address

139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.

In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

Ballard, 7:15-53.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Ballard, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> A general purpose personal computer typically has many interactive application computer programs installed. A user is able to start-up multiple programs. With regard to an interactive computer program, the term "launch time", as used herein, means the time from when a processor

> receives a command to start the computer program until the time that the computer program is ready to accept input commands (e.g., user interface commands, batch-entry commands). The term "launch" as used herein means the process performed during the launch time to start up the computer program and get the computer ready to accept input commands for the computer progra[m].

Ballard, 1:38-50

> According to another aspect of the invention, the launch access log is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.
>
> A launch sequence as used herein means the sequence of steps executed by the computer system during the launch time which pertain to starting up a given computer program and getting the computer ready to accept input commands for the computer program. A launch sequence is executed for a computer program. Different computer programs have different launch sequences. Steps included in a launch sequence include copying portions of the computer program being launched from the secondary storage device to the primary storage device. Other steps may include allocating a port or device to serve as an input source and/or output receptor. The method of this invention for accelerating a program launch is directed to improving the speed for accessing the secondary

storage device during a launch sequence.

Ballard, 3:40-67

> Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.
>
> In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

Ballard, 7:15-53.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Ballard, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> A general purpose personal computer typically has many interactive application computer programs installed. A user is able to start-up multiple programs. With regard to an interactive computer program, the term "launch time", as used herein, means the time from when a processor receives a command to start the computer program until the time that the computer program is ready to accept input commands (e.g., user interface commands, batch-entry commands). The term "launch" as used herein means the process performed during the launch time to start up the

computer program and get the computer ready to accept input commands for the computer program.

It is common for an application program for a personal computer to be stored in multiple files on the secondary storage device. There often is an executable file, a preferences file and many other files. Some programs include a data base file or a default data file. During a launch of the program multiple files are opened and select portions are moved from the secondary storage device into the primary storage device. When purchasing a computer program the packaging often specifies the amount of RAM address space (i.e., primary storage device address space) required to be allocated to the program while active. By active it is meant that the program has been launched and is currently processing or is currently able to accept input commands.

Ballard, 1:38-63

The processor 12 serves to execute an operating system and one or more application computer programs. In some embodiments there are multiple processors for executing the operating system and application programs. System utilities and/or operating system extension programs also are executed according to some computer system 10 embodiments. Conventional operating systems include DOS, Windows, Windows NT, MacOS, OS/2 and various UNIX-based operating systems. The display device 18 and input devices 20 enable interaction between a user and the computer system 10. The computer system 10 in the process of executing the operating system and zero or more computer programs defines an operating environment for a user to interact with the computer, operating system and executing computer program. The display device 18 serves as an output device. Exemplary display devices include a CRT monitor or flat panel display. The user inputs commands and data to the computer system 10 via the input devices. Exemplary input devices include a keyboard, a pointing device and a clicking device. Data also is input to the computer via transportable disks or through I/O ports (not shown).

Ballard, 3:10-30.

Log File System Activity

Once the launch of a computer program is detected and a log file is opened, all file system activity is monitored. Specifically, for each operating system call to the file system the call is analyzed to determine to which application being launched, if any, does the call pertain. Exemplary operating system calls to the file system are OPEN, READ, WRITE, and CLOSE. Referring to FIG. 4 routine 82 is entered at step 84

when both file system activity is detected and logging is enabled. If the call pertains to a computer program being launched, then an entry is appended to the appropriate log file (step 86). The routine 82 then returns at step 88.

The log entry includes a file identifier, the logical address(es) specified in the call and the time of access (e.g., system time; index value). Alternatively, the physical memory address(es) corresponding to the logical address(es) are stored in the log entry. In some embodiments, the operating system has already caused the physical addresses to be generated. If not, then the physical addresses are derived from the logical addresses using the operating system's file allocation table to translate the logical address into the physical address.

A logical address (also referred to as a virtual address) is the address which the computer program uses to access memory. A memory management unit translates this address into a physical address before the actual memory is read or written. A physical address is a memory location on the secondary storage device 16.

Ballard, 5:1-37

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Ballard, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.3, and 1.4 above.	

4. The method of claim 1, further comprising updating the list of boot data.	Ballard, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claim 1.1 above.

See also

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> According to another aspect of the invention, the launch access log is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

> Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are

located at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.

In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

The modified log file next is sorted at step 96. In one embodiment the log entries are grouped according to the file. Each access resulting in a log entry specifies a file and an address. All entries for a given file are grouped together, then arranged within the group by logical address. The groups are arranged chronologically according to which file is specified first in the log file. Alternatively other criteria for ordering the groups is used. In an alternative embodiment instead of grouping the entries by file, all the log entries are sorted according to physical address to optimize access time to the secondary storage device 16. In one embodiment the log entries define a queue processed according to the methods disclosed in U.S. patent application Ser. No. 08/656,372 filed May 31, 1996 for "Estimating Access Time for Hard Drive I/O Requests." The contents of such application are incorporated herein by

reference and made a part hereof. The log entries are rearranged in an order to optimize access time as described therein.

Ballard, 7:15-8:4.

See also Ballard, 5:1-7:14, Figs. 4-6

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Ballard, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Ballard, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Ballard, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Ballard, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.2, 1.3, and 1.4 above.	

7.2 a cache memory device; and;	Ballard, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Ballard, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Ballard, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Ballard, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Ballard, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Appendix A3
Invalidity of U.S. Patent 7,181,608 based on Ballard

Ballard, Abstract

> A general purpose personal computer typically has many interactive application computer programs installed. A user is able to start-up multiple programs. With regard to an interactive computer program, the term "launch time", as used herein, means the time from when a processor receives a command to start the computer program until the time that the computer program is ready to accept input commands (e.g., user interface commands, batch-entry commands). The term "launch" as used herein means the process performed during the launch time to start up the computer program and get the computer ready to accept input commands for the computer program.
>
> It is common for an application program for a personal computer to be stored in multiple files on the secondary storage device. There often is an executable file, a preferences file and many other files. Some programs include a data base file or a default data file. During a launch of the program multiple files are opened and select portions are moved from the secondary storage device into the primary storage device. When purchasing a computer program the packaging often specifies the amount of RAM address space (i.e., primary storage device address space) required to be allocated to the program while active. By active it is meant that the program has been launched and is currently processing or is currently able to accept input commands.

Ballard, 1:38-63

> The processor 12 serves to execute an operating system and one or more application computer programs. In some embodiments there are multiple processors for executing the operating system and application programs. System utilities and/or operating system extension programs also are executed according to some computer system 10 embodiments. Conventional operating systems include DOS, Windows, Windows NT, MacOS, OS/2 and various UNIX-based operating systems. The display device 18 and input devices 20 enable interaction between a user and the computer system 10. The computer system 10 in the process of executing the operating system and zero or more computer programs defines an operating environment for a user to interact with the computer, operating system and executing computer program. The display device 18 serves as an output device. Exemplary display devices include a CRT monitor or flat panel display. The user inputs commands and data to the computer system 10 via the input devices. Exemplary input devices include a keyboard, a pointing device and a clicking device. Data also is input to

the computer via transportable disks or through I/O ports (not shown).

Ballard, 3:10-30.

Log File System Activity

Once the launch of a computer program is detected and a log file is opened, all file system activity is monitored. Specifically, for each operating system call to the file system the call is analyzed to determine to which application being launched, if any, does the call pertain. Exemplary operating system calls to the file system are OPEN, READ, WRITE, and CLOSE. Referring to FIG. 4 routine 82 is entered at step 84 when both file system activity is detected and logging is enabled. If the call pertains to a computer program being launched, then an entry is appended to the appropriate log file (step 86). The routine 82 then returns at step 88.

The log entry includes a file identifier, the logical address(es) specified in the call and the time of access (e.g., system time; index value). Alternatively, the physical memory address(es) corresponding to the logical address(es) are stored in the log entry. In some embodiments, the operating system has already caused the physical addresses to be generated. If not, then the physical addresses are derived from the logical addresses using the operating system's file allocation table to translate the logical address into the physical address.

A logical address (also referred to as a virtual address) is the address which the computer program uses to access memory. A memory management unit translates this address into a physical address before the actual memory is read or written. A physical address is a memory location on the secondary storage device 16.

Ballard, 5:1-37

9.1 maintaining a list of application data associated with an application program;	Ballard, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Ballard, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Ballard, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> A general purpose personal computer typically has many interactive

> application computer programs installed. A user is able to start-up multiple programs. With regard to an interactive computer program, the term "launch time", as used herein, means the time from when a processor receives a command to start the computer program until the time that the computer program is ready to accept input commands (e.g., user interface commands, batch-entry commands). The term "launch" as used herein means the process performed during the launch time to start up the computer program and get the computer ready to accept input commands for the computer program.
>
> It is common for an application program for a personal computer to be stored in multiple files on the secondary storage device. There often is an executable file, a preferences file and many other files. Some programs include a data base file or a default data file. During a launch of the program multiple files are opened and select portions are moved from the secondary storage device into the primary storage device. When purchasing a computer program the packaging often specifies the amount of RAM address space (i.e., primary storage device address space) required to be allocated to the program while active. By active it is meant that the program has been launched and is currently processing or is currently able to accept input commands.

Ballard, 1:38-63

> The processor 12 serves to execute an operating system and one or more application computer programs. In some embodiments there are multiple processors for executing the operating system and application programs. System utilities and/or operating system extension programs also are executed according to some computer system 10 embodiments. Conventional operating systems include DOS, Windows, Windows NT, MacOS, OS/2 and various UNIX-based operating systems. The display device 18 and input devices 20 enable interaction between a user and the computer system 10. The computer system 10 in the process of executing the operating system and zero or more computer programs defines an operating environment for a user to interact with the computer, operating system and executing computer program. The display device 18 serves as an output device. Exemplary display devices include a CRT monitor or flat panel display. The user inputs commands and data to the computer system 10 via the input devices. Exemplary input devices include a keyboard, a pointing device and a clicking device. Data also is input to the computer via transportable disks or through I/O ports (not shown).

Ballard, 3:10-30.

Log File System Activity

Once the launch of a computer program is detected and a log file is opened, all file system activity is monitored. Specifically, for each operating system call to the file system the call is analyzed to determine to which application being launched, if any, does the call pertain. Exemplary operating system calls to the file system are OPEN, READ, WRITE, and CLOSE. Referring to FIG. 4 routine 82 is entered at step 84 when both file system activity is detected and logging is enabled. If the call pertains to a computer program being launched, then an entry is appended to the appropriate log file (step 86). The routine 82 then returns at step 88.

The log entry includes a file identifier, the logical address(es) specified in the call and the time of access (e.g., system time; index value). Alternatively, the physical memory address(es) corresponding to the logical address(es) are stored in the log entry. In some embodiments, the operating system has already caused the physical addresses to be generated. If not, then the physical addresses are derived from the logical addresses using the operating system's file allocation table to translate the logical address into the physical address.

A logical address (also referred to as a virtual address) is the address which the computer program uses to access memory. A memory management unit translates this address into a physical address before the actual memory is read or written. A physical address is a memory location on the secondary storage device 16.

Ballard, 5:1-37

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Ballard, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

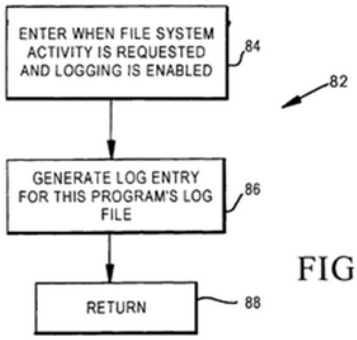


FIG. 4

Ballard, Fig. 4

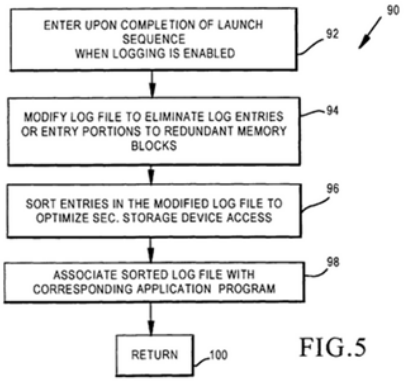


FIG.5

Ballard, Fig. 5

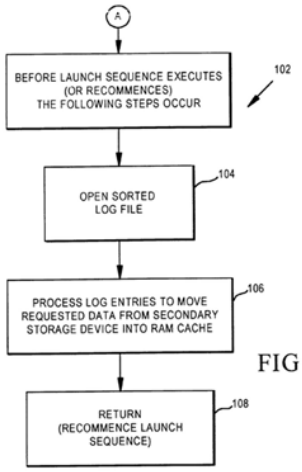


FIG 6

Ballard, Fig. 6

> It is common for an application program for a personal computer to be stored in multiple files on the secondary storage device. There often is an executable file, a preferences file and many other files. Some programs include a data base file or a default data file. During a launch of the program multiple files are opened and select portions are moved from the secondary storage device into the primary storage device. When purchasing a computer program the packaging often specifies the amount of RAM address space (i.e., primary storage device address space) required to be allocated to the program while active. By active it is meant that the program has been launched and is currently processing or is currently able to accept input commands.

Ballard Claim 10

"The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device"

Ballard, 1:51-63

> According to another aspect of the invention, the launch access log is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.
>
> A launch sequence as used herein means the sequence of steps executed by the computer system during the launch time which pertain to starting up a given computer program and getting the computer ready to accept input commands for the computer program. A launch sequence is executed for a computer program. Different computer programs have different launch sequences. Steps included in a launch sequence include copying portions of the computer program being launched from the secondary storage device to the primary storage device. Other steps may include allocating a port or device to serve as an input source and/or output receptor. The method of this invention for accelerating a program launch is directed to improving the speed for accessing the secondary storage device during a launch sequence.

Ballard, 3:40-67

> The log file corresponds to a specific computer program--the one being launched that triggered such log file to be created. When multiple programs are being launched at the same time, a log file is created for each such program. The interrupt service routine then sets a flag at step 78 to indicate that logging is enabled for such program. At step 80 the program returns. If the trigger for calling the routine 70 was for a secondary storage device access, then the steps at FIG. 4 also are

performed before returning.

Log File System Activity

Once the launch of a computer program is detected and a log file is opened, all file system activity is monitored. Specifically, for each operating system call to the file system the call is analyzed to determine to which application being launched, if any, does the call pertain. Exemplary operating system calls to the file system are OPEN, READ, WRITE, and CLOSE. Referring to FIG. 4 routine 82 is entered at step 84 when both file system activity is detected and logging is enabled. If the call pertains to a computer program being launched, then an entry is appended to the appropriate log file (step 86). The routine 82 then returns at step 88.

The log entry includes a file identifier, the logical address(es) specified in the call and the time of access (e.g., system time; index value). Alternatively, the physical memory address(es) corresponding to the logical address(es) are stored in the log entry. In some embodiments, the operating system has already caused the physical addresses to be generated. If not, then the physical addresses are derived from the logical addresses using the operating system's file allocation table to translate the logical address into the physical address.

A logical address (also referred to as a virtual address) is the address which the computer program uses to access memory. A memory management unit translates this address into a physical address before the actual memory is read or written. A physical address is a memory location on the secondary storage device 16.

Ballard, 5:1-37

Detect Launch Completion

When completion of a launch sequence occurs and logging is enabled, then routine 90 is executed. Referring to FIG. 5, the routine enters at step 92. Conventional operating systems have a specific function that is called when a program is ready for normal execution. Under the Macintosh operating system, the function "Get Next Event" is called. For a computer program running under such operating system, such function is called by the computer program when the launch sequence is complete. According to one embodiment of this invention, step 92 is implemented by an interrupt service routine which is called whenever a computer program calls such function. The interrupt service routine clears the logging

enabled flag for the corresponding application program.

In an alternative embodiment, access activity to the secondary storage device 16 is monitored to determine when activity has ceased for a threshold length of time (e.g., 3 seconds). Alternatively or in addition activity is monitored to determine when activity has gone below a threshold data throughput rate (e.g., 50 kilobytes per second) for a threshold period of time (e.g., 5 seconds). When there is insufficient activity for such threshold time, then the program launch is considered to be complete. The routine 90 then is executed.

Modify and Sort Log File

Once the launch sequence is determined to have been completed, then the log entry order is re-organized. The purpose is to eliminate redundant accesses to the same memory block and to optimize access time for the secondary storage device. If accesses occur faster, then the launch time (i.e., time elapsed from start to finish of launch sequence is less) is reduced. Thus, the launch of the computer program is accelerated.

Referring to FIG. 5, at step 94 the log file is processed to eliminate log entries or log entry portions to redundant memory blocks.

Ballard, 5:55-6:23

Note in this example that the log includes redundant entries. Entry h2 is a subset of entry d. Entry i2 is the same as entry f. Entry l2 is the same as entry a. It is expected that a redundant access request results from a computer program launch sequence access specifying a different address in the same block as previously accessed.

Frequently the log file will include an entry specifying a single address. Even though only one address is specified, an entire memory block will be accessed (read or written). This is because the minimum file allocation unit is a memory block cluster. Thus, the smallest portion of the computer program than can be accessed is the cluster size (i.e., cluster size). Entries in the log file are tested at step 94 to identify any redundant accesses to portions of the computer program. To determine whether a later entry is an access to a redundant portion of the computer program, the cluster address for the access is identified. The cluster address is either stored in the log file or is derived from the address stored in the log file. For a FAT drive the cluster size is stored in the drive's boot sector. The boot sector includes information about the layout of the file system used for the drive partition. Following the boot sector are several reserved sectors. Following the reserved sectors is the file allocation table (FAT).

> Following the FAT are one or more backup copies of the FAT. Following the backup copies is the root directory. The size of the root directory is specified in the boot sector. After the root directory are the user's file and directory area. This is the area divided into clusters. Thus, from the information in the boot sector the starting address of cluster space is determined, along with the size of a cluster. Thus, the address boundaries for each cluster are known.
>
> The address for any given cluster x is given by Ax+B, where A and B are constants determined from the boot partition. Thus, for any given file system call the cluster boundaries for such address are determinable. The log file stores either the accessed address, the cluster number (i.e., x) or the cluster address (e.g., start address, end address or some other identifying address) for each cluster accessed.

Ballard, 6:44-7:14

> Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.
>
> In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of

> by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.
>
> Ballard, 7:15-53.
>
> *See also* Ballard, Tables B-C

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Ballard, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> It is common for an application program for a personal computer to be stored in multiple files on the secondary storage device. There often is an executable file, a preferences file and many other files. Some programs include a data base file or a default data file. During a launch of the program multiple files are opened and select portions are moved from the secondary storage device into the primary storage device. When purchasing a computer program the packaging often specifies the amount of RAM address space (i.e., primary storage device address space) required to be allocated to the program while active. By active it is meant that the program has been launched and is currently processing or is currently able to accept input commands.

Ballard, 1:51-63

> According to another aspect of the invention, the launch access log is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining

> accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.
>
> A launch sequence as used herein means the sequence of steps executed by the computer system during the launch time which pertain to starting up a given computer program and getting the computer ready to accept input commands for the computer program. A launch sequence is executed for a computer program. Different computer programs have different launch sequences. Steps included in a launch sequence include copying portions of the computer program being launched from the secondary storage device to the primary storage device. Other steps may include allocating a port or device to serve as an input source and/or output receptor. The method of this invention for accelerating a program launch is directed to improving the speed for accessing the secondary storage device during a launch sequence.

Ballard, 3:40-67

> Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is

already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.

In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

Ballard, 7:15-53.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Ballard, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Ballard, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1.3 and 1.4 above.

See also

> The secondary storage device 16 serves as a permanent storage memory for one or more computer programs 24 to be executed by the processor 12. The secondary storage device 16 also stores data files for use with the application computer programs. Exemplary secondary storage devices include a hard disk drive, floppy disk drive, CD-ROM drive, bernoulli disk drive or other drive system for accessing permanent or replaceable disks, such as floppy disks, magnetic disks, magneto-optical disks, or optical disks.

Ballard, 3:31-39

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Ballard, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Ballard, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.3, 1.4, and 15 above.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Ballard, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Ballard, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Ballard, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Ballard, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Ballard, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Ballard, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Ballard, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Ballard, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Ballard, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Ballard, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Ballard, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Ballard, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Ballard, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Ballard, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Ballard, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Ballard, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Ballard, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Ballard, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 15 and 27 above	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Ballard, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 16 and 27 above	

Appendix A4
Invalidity of U.S. Patent 7,181,608 based on Feigenbaum

U.S. Patent No. 5,307,497 to Feigenbaum ("Feigenbaum") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A4
Invalidity of U.S. Patent 7,181,608 based on Feigenbaum

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Feigenbaum, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> This invention relates to the field of data processing, and, more particularly, to a method and apparatus for loading a personal computer disk operating system (DOS) from a read only memory (ROM) using an installable file system (IFS) interface.

Feigenbaum, 1:7-12

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

> One of the objects of the invention is to provide a ROM based DOS that rapidly tests, boots, and initializes a personal computer and appears to the user to "instantly" start up when the computer is turned on.

Feigenbaum, 3:30-33

> A further object is to provide a ROM based DOS which uses an IFS that is accessible at the system level, as opposed to the device level, to rapidly load DOS programs from ROM into RAM for execution from such RAM.

Feigenbaum, 3:38-42

> Briefly, in accordance with the subject invention, bootable DOS programs are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, 3:53-59

> When switch 35 is initially turned on, display 30 warms up within a relatively short period and the ROM based booting and system initialization, described in detail below, occurs within the period required for the display to warm up so that at the end of such period, the display screen becomes visible to the user to give the appearance of an instant startup.

Feigenbaum, 4:63-5:2

> In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code.
>
> Feigenbaum, 10:18-33

1.1 maintaining a list of boot data used for booting a computer system;	Feigenbaum, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by

> storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

> System 10 is further provided with a DOS 32 that comprises two major portions, ROM DOS 34 and DISK DOS 36 respectively stored in ROM 16 and disk 24. ROM DOS 34 includes the programs that are booted into the RAM when the system is initially powered on or when the system is reset, which programs provide the minimum level of operating system support to make system 10 operational to the user. The remaining portions of DOS, i.e. those programs and files that together form a complete DOS, are included in DISK DOS 36. DISK DOS 36 further includes DOS programs similar to those in ROM DOS so that the system can be booted up and operated from disk 24 at the option of the user, as described in more detail hereafter. Also stored in ROM 16 are a basic input/output system (BIOS) 38 and a power on self test (POST) routine 40. When system 10 is first turned on, or when it is restarted, POST 40 first performs the test and upon successful completion, it boots or loads a portion of DOS and transfers control to it in the manner described in detail below. Except for ROM DOS 34, the system as thus far described is constructed and operates in accordance with known principles of the prior art.

Feigenbaum, 5:3-25

1.2 initializing a central processing unit of the computer system;	Feigenbaum, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> Referring now to the drawings, and first to FIG. 1, the invention is embodied in a personal computer system 10, and resides in the manner in which such system is programmed and operated. It is to be appreciated that such computers are complex and include many components and data processing devices, such as device controllers and adapters, which have been omitted from the drawings for simplicity of illustration. The description provided herein is limited to only those items which are useful in understanding the invention. System 10 includes a microprocessor 12, such as an Intel 80286 microprocessor, which is commercially available and functions in a known manner to execute programs stored in a RAM 14 and a ROM 16. Such ROM preferably comprises a plurality of ROM units which together form ROM 16 and provide sufficient storage capacity for all of the stored information described in more detail below.

Feigenbaum, 4:10-27

> With reference to FIG. 2, a memory map 39 is illustrated based on the full address space of sixteen megabytes (MB) using the twenty-four bit addressing capability of microprocessor 12. The lowest 640 kilobytes (KB) are assigned to RAM 14 and form a storage area, known as the DOS address space, for containing DOS programs in the lowest portion thereof and application programs in the upper or top portion thereof. Addresses immediately below the one megabyte (MB) region are assigned to that portion of ROM 20 containing POST 38 and BIOS 40. A video buffer occupies the space immediately above the DOS address space

> beginning at hex address A0000 H. ROM DOS 34 occupies a 256KB region at the top of the 16 MB address space beginning at hex address FC0000 H. Thus, the lowest one MB region is assigned to the same addresses and functions as such region is used in the prior art, while ROM DOS 34 is assigned to an address space which is not immediately addressable by DOS nor application programs running in the DOS address space. The remaining regions are unused memory addresses. Further, the low 1MB address range is accessible in the real mode of operation of microprocessor 12, while the address range above the real mode range is accessible in protected mode.

Feigenbaum, 5:26-49

> The steps are performed by microprocessor 12 operating under program control to control the operation of the various components within system 10.

Feigenbaum, 8:34-37

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Feigenbaum, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy diskette can

be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

System 10 is further provided with a DOS 32 that comprises two major portions, ROM DOS 34 and DISK DOS 36 respectively stored in ROM 16 and disk 24. ROM DOS 34 includes the programs that are booted into the RAM when the system is initially powered on or when the system is reset, which programs provide the minimum level of operating system support to make system 10 operational to the user. The remaining portions of DOS, i.e. those programs and files that together form a complete DOS, are included in DISK DOS 36. DISK DOS 36 further includes DOS programs similar to those in ROM DOS so that the system can be booted up and operated from disk 24 at the option of the user, as described in more detail hereafter. Also stored in ROM 16 are a basic input/output system (BIOS) 38 and a power on self test (POST) routine 40. When system 10 is first turned on, or when it is restarted, POST 40 first performs the test and upon successful completion, it boots or loads a portion of DOS and transfers control to it in the manner described in detail below. Except for ROM DOS 34, the system as thus far described is constructed and operates in accordance with known principles of the prior art.

Feigenbaum, 5:3-25

FIG. 3 also illustrates another facet of the invention which is that the data and programs of ROM DOS 34 are stored in ROM 16 in a file system using a packed format. In contrast, any data and files stored on disk 24, including those in RAM DOS 36, are stored in the conventional DOS FAT file system in which information is stored in clusters and sectors. In the FAT file system, if a particular program doesn't have the same number of bytes as are in a sector or cluster, space is wasted. On the average, about one half the number of bytes in a cluster or in a sector are wasted for each file. Since ROM units are more expensive than disk storage, any wasted space is inefficient. Thus, in ROM 16, the ROM DOS 34 files are stored beginning at a segment address, each segment being sixteen bytes. Each succeeding file begins immediately at the next segment location following the end of a preceding file so that on average only eight bytes per file would be

wasted, such number being far less than the average waste in a FAT file system. By way of example, suppose file 50 begins at ROM segment X and ends at Y. Then, the succeeding file 52 begins at segment Y+n, where n is less than sixteen. The other files are similarly stored.

Feigenbaum, 6:60-7:15

When system 11 is turned on, the general sequence of operations that occur are BIOSPOST 64, ROMBOOT 65, IBMINIT 66, SYSINIT 67, and COMMAND 68. BIOSPOST 64 first performs a power on self test 70 which is the same as or similar to what is done within the prior art. Upon successful completion of such test, step 72 then checks flag 58 to see if the fast boot process of the invention has been selected. If it has not, then the system would boot from disk 24 in accordance with the prior art. If such selection was made, then step 74 copies the first 512 bytes of ROM DOS 34 into RAM 14 at the top of such memory, the bytes thus copied including ROMBOOT 48. Control is then passed via step 75 to ROMBOOT 65 by jumping from POST to instruction 44 and then to the start of ROMBOOT program 48.

Feigenbaum, 8:37-52

ROMBOOT 65 first loads IBMBIO.COM 50, including RAM LOADER 51, from ROM 16 into RAM 14 by step 76. IBMBIO.COM includes two code segments, IBMINIT and SYSINIT. Step 78 then transfers control to IBMINIT 66. Step 80 initializes the system device drivers and hardware. Step 81 provides or sets up a hook into interrupt handler INT 2BH for the RAM LOADER. Step 82 relocates the initialization routine of IBMBIO.COM 50 to predetermined locations at the top of the DOS address space in RAM 16. Next, step 84 transfers control to SYSINIT 67 which first, by step 86, loads IBMDOS.COM 52 from ROM 16 into RAM 14. Step 88 then executes the initialization routine of IBMDOS.COM. Afterwards, step 90 attaches the IFS Handler as the next available drive which is drive D: because system 10 includes a fixed disk drive. Next, step 91 selects the ROM drive created by the IFS handler, as the defualt drive.

Feigenbaum, 8:53-9:2

In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of

changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code.

Feigenbaum, 10:18-33

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Feigenbaum, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the

> system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> System 10 is further provided with a DOS 32 that comprises two major portions, ROM DOS 34 and DISK DOS 36 respectively stored in ROM 16 and disk 24. ROM DOS 34 includes the programs that are booted into the RAM when the system is initially powered on or when the system is reset, which programs provide the minimum level of operating system support to make system 10 operational to the user. The remaining portions of DOS, i.e. those programs and files that together form a complete DOS, are included in DISK DOS 36. DISK DOS 36 further includes DOS programs similar to those in ROM DOS so that the system can be booted up and operated from disk 24 at the option of the user, as described in more detail hereafter. Also stored in ROM 16 are a basic input/output system (BIOS) 38 and a power on self test (POST) routine 40. When system 10 is first turned on, or when it is restarted, POST 40 first performs the test and upon successful completion, it boots or loads a portion of DOS and transfers control to it in the manner described in detail below. Except for ROM DOS 34, the system as thus far described is constructed and operates in accordance with known principles of the prior art.

Feigenbaum, 5:3-25

> When system 10 is first setup, it is desireable to customize the system in a manner similar to that in which IBM PS/2 systems are configured. During the course of such customization the user has to make decisions and define what features are to be used. The primary decision pertinent to the invention is whether to boot the system from ROM, in accordance with the invention, or to boot the system from disk 24 in the manner of the prior art. The selection of the option is then coded into a flag 58 which is stored during customization in CMOS RAM 20. Other options for the user are whether the system will be started by using the ROM based CONFIG.SYS 62 and/or AUTOEXEC.BAT 61, or the user defined CONFIG.SYS 102 or AUTOEXEC.BAT 102 files stored on disk 24. Flag 58 will also be set to reflect such other options. Default values are provided for such options so that in the absence of selecting other options, the system will automatically boot from ROM 16 and process the AUTOEXEC.BAT and CONFIG.SYS files therein.

Feigenbaum, 7:15-35

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Feigenbaum, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

> With reference to FIG. 2, a memory map 39 is illustrated based on the full address space of sixteen megabytes (MB) using the twenty-four bit addressing capability of microprocessor 12. The lowest 640 kilobytes (KB) are assigned to RAM 14 and form a storage area, known as the DOS address space, for containing DOS programs in the lowest portion thereof and application programs in the upper or top portion thereof. Addresses immediately below the one megabyte (MB) region are assigned to that portion of ROM 20 containing POST 38 and BIOS 40. A video buffer occupies the space immediately above the DOS address space beginning at hex address A0000 H. ROM DOS 34 occupies a 256KB region at the top of the 16 MB address space beginning at hex address FC0000 H. Thus, the lowest one MB region is assigned to the same addresses and functions as such region is used in the prior art, while ROM DOS 34 is assigned to an address space which is not immediately addressable by DOS nor application programs running in the DOS address space. The remaining regions are unused memory addresses. Further, the low 1MB address range is accessible in the real mode of operation of microprocessor 12, while the address range above the real mode range is accessible in protected mode.

Feigenbaum, 5:26-49

> IBMDOS.COM provides application support.

Feigenbaum, 6:11

> RAM LOADER 51, IFS HANDLER 53, and CHECKC.COM 63 are new with the invention, while the remaining programs stored in ROM 16 perform functions previously commercially available. ROM 16 may optionally be of a larger size to store additional programs such as a code page switching program, a program providing extended keyboard layout, support for other national languages, etc. It may also contain alternative AUTOEXEC.BAT and CONFIG.SYS files 102 and 104 oriented to the use of such additional programs. The size of ROM 16 may thus vary dependent upon the size of and how many additional support programs are stored. Preferably, those items shown in FIG. 3 are stored in a 128K ROM unit and the additional programs are stored in a second 128 K ROM unit.

Feigenbaum, 6:45-59

> FIG. 4 illustrates the relative position and use of IFS HANDLER 53. Within the prior art, an application program 110 can be thought of as sitting on top of FAT file system 112, which in turn sits on top of disk/diskette driver 114 which overlies the disk 24 containing the actual files and data. Application 110 uses interrupt INT 21H to access the file system through IBMDOS.COM, which acts through IBMBIO.COM to access the disk. In accordance with the invention, IFS HANDLER 53 is interposed at the file system level in the INT 21H process and decides in step 116 whether the desired file system access is to be made to ROM DISK D:. If not, the system proceeds as in the prior art to access the FAT file system. If the desired access is to be made to ROM DISK D:, it is done through the ROM file system 118 and RAM LOADER 51 by programs IBMDOS.COM 52 and IBMBIO.COM 50. The term "ROM DISK" is defined hereby to mean a ROM unit containing images of files storable on a hard disk or floppy diskette, which images are stored in a packed format in a ROM file system that is different from the FAT file system used to store files on a disk or diskette.

Feigenbaum, 9:39-60

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Feigenbaum, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1.3, and 1.4 above.

> Referring now to the drawings, and first to FIG. 1, the invention is embodied in a personal computer system 10, and resides in the manner in which such system is programmed and operated. It is to be appreciated that such computers are complex and include many components and data processing devices, such as device controllers and adapters, which have been omitted from the drawings for simplicity of illustration. The description provided herein is limited to only those items which are useful in understanding the invention. System 10 includes a microprocessor 12, such as an Intel 80286 microprocessor, which is commercially available and functions in a known manner to execute programs stored in a RAM 14 and a ROM 16. Such ROM preferably comprises a plurality of ROM units which together form ROM 16 and provide sufficient storage capacity for all of the stored information described in more detail below.

Feigenbaum, 4:10-27

> The steps are performed by microprocessor 12 operating under program control to control the operation of the various components within system 10.

Feigenbaum, 8:34-37

4. The method of claim 1, further comprising updating the list of boot data.	Feigenbaum, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claim 1.1 above.

> Another solution that might occur would be to create the ROM within the DOS address space, or within the first one megabyte of memory mapped space, and then load or copy DOS into a RAM (random access memory) within such address space to occupy and execute from the same space in RAM and operate in the same manner as DOS currently does. The disadvantage of such a solution is that two copies of DOS would then exist in such limited address space, one copy being the unalterable ROM DOS and the other copy being the alterable one that is stored in and executed from the RAM. Having two copies of essentially the same program in such a limited address space would not be efficient use of memory nor acceptable by many users.

Feigenbaum, 1:58-2:3

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Feigenbaum, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Feigenbaum, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Feigenbaum, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Feigenbaum, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

> Referring now to the drawings, and first to FIG. 1, the invention is embodied in a personal computer system 10, and resides in the manner in which such system is programmed and operated. It is to be appreciated that such computers are complex and include many components and data processing devices, such as device controllers and adapters, which have been omitted from the drawings for simplicity of illustration. The description provided herein is limited to only those items which are useful in understanding the invention. System 10 includes a microprocessor 12, such as an Intel 80286 microprocessor, which is commercially available and functions in a known manner to execute programs stored in a RAM 14 and a ROM 16. Such ROM preferably comprises a plurality of ROM units which together form ROM 16 and provide sufficient storage capacity for all of the stored information described in more detail below.

Feigenbaum, 4:10-27

> The steps are performed by microprocessor 12 operating under program control to control the operation of the various components within system 10.

Feigenbaum, 8:34-37

7.2 a cache memory device; and;	Feigenbaum, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Feigenbaum, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Feigenbaum, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Feigenbaum, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Feigenbaum, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

> With reference to FIG. 2, a memory map 39 is illustrated based on the full address space of sixteen megabytes (MB) using the twenty-four bit addressing capability of microprocessor 12. The lowest 640 kilobytes (KB) are assigned to RAM 14 and form a storage area, known as the DOS address space, for containing DOS programs in the lowest portion thereof and application programs in the upper or top portion thereof. Addresses immediately below the one megabyte (MB) region are assigned to that portion of ROM 20 containing POST 38 and BIOS 40. A video buffer occupies the space immediately above the DOS address space beginning at hex address A0000 H. ROM DOS 34 occupies a 256KB region at the top of the 16 MB address space beginning at hex address FC0000 H. Thus, the lowest one MB region is assigned to the same addresses and functions as such region is used in the prior art, while ROM DOS 34 is assigned to an address space which is not immediately addressable by DOS nor application programs running in the DOS address space. The remaining regions are unused memory addresses. Further, the low 1MB address

range is accessible in the real mode of operation of microprocessor 12, while the address range above the real mode range is accessible in protected mode.

Feigenbaum, 5:26-49

IBMDOS.COM provides application support.

Feigenbaum, 6:11

RAM LOADER 51, IFS HANDLER 53, and CHECKC.COM 63 are new with the invention, while the remaining programs stored in ROM 16 perform functions previously commercially available. ROM 16 may optionally be of a larger size to store additional programs such as a code page switching program, a program providing extended keyboard layout, support for other national languages, etc. It may also contain alternative AUTOEXEC.BAT and CONFIG.SYS files 102 and 104 oriented to the use of such additional programs. The size of ROM 16 may thus vary dependent upon the size of and how many additional support programs are stored. Preferably, those items shown in FIG. 3 are stored in a 128K ROM unit and the additional programs are stored in a second 128 K ROM unit.

Feigenbaum, 6:45-59

FIG. 4 illustrates the relative position and use of IFS HANDLER 53. Within the prior art, an application program 110 can be thought of as sitting on top of FAT file system 112, which in turn sits on top of disk/diskette driver 114 which overlies the disk 24 containing the actual files and data. Application 110 uses interrupt INT 21H to access the file system through IBMDOS.COM, which acts through IBMBIO.COM to access the disk. In accordance with the invention, IFS HANDLER 53 is interposed at the file system level in the INT 21H process and decides in step 116 whether the desired file system access is to be made to ROM DISK D:. If not, the system proceeds as in the prior art to access the FAT file system. If the desired access is to be made to ROM DISK D:, it is done through the ROM file system 118 and RAM LOADER 51 by programs IBMDOS.COM 52 and IBMBIO.COM 50. The term "ROM DISK" is defined hereby to mean a ROM unit containing images of files storable on a hard disk or floppy diskette, which images are stored in a packed format in a ROM file system, that is different from the FAT file system used to store files on a disk or diskette.

Feigenbaum, 9:39-60

Appendix A4
Invalidity of U.S. Patent 7,181,608 based on Feigenbaum

9.1 maintaining a list of application data associated with an application program;	Feigenbaum, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Feigenbaum, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Feigenbaum, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1.4, 2, and 8 above.

> With reference to FIG. 2, a memory map 39 is illustrated based on the full address space of sixteen megabytes (MB) using the twenty-four bit addressing capability of microprocessor 12. The lowest 640 kilobytes (KB) are assigned to RAM 14 and form a storage area, known as the DOS address space, for containing DOS programs in the lowest portion thereof and application programs in the upper or top portion thereof. Addresses immediately below the one megabyte (MB) region are assigned to that portion of ROM 20 containing POST 38 and BIOS 40. A video buffer occupies the space immediately above the DOS address space beginning at hex address A0000 H. ROM DOS 34 occupies a 256KB region at the top of the 16 MB address space beginning at hex address FC0000 H. Thus, the lowest one MB region is assigned to the same addresses and functions as such region is used in the prior art, while ROM DOS 34 is assigned to an address space which is not immediately addressable by DOS nor application programs running in the DOS address space. The remaining regions are unused memory addresses. Further, the low 1MB address range is accessible in the real mode of operation of microprocessor 12, while the address range above the real mode range is accessible in protected mode.

Feigenbaum, 5:26-49

> IBMDOS.COM provides application support.

Feigenbaum, 6:11

> RAM LOADER 51, IFS HANDLER 53, and CHECKC.COM 63 are new with the invention, while the remaining programs stored in ROM 16 perform functions previously commercially available. ROM 16 may optionally be of a larger size to store additional programs such as a code page switching program, a program providing extended keyboard layout, support for other national languages, etc. It may also contain alternative AUTOEXEC.BAT and CONFIG.SYS files 102 and 104 oriented to the use of such additional programs. The size of ROM 16 may thus vary dependent upon the size of and how many additional support programs are stored. Preferably, those items shown in FIG. 3 are stored in a 128K ROM unit and the additional programs are stored in a second 128 K ROM unit.

Feigenbaum, 6:45-59

> FIG. 4 illustrates the relative position and use of IFS HANDLER 53. Within the prior art, an application program 110 can be thought of as sitting on top of FAT file system 112, which in turn sits on top of disk/diskette driver 114 which overlies the disk 24 containing the actual files and data. Application 110 uses interrupt INT 21H to access the file system through IBMDOS.COM, which acts through IBMBIO.COM to access the disk. In accordance with the invention, IFS HANDLER 53 is interposed at the file system level in the INT 21H process and decides in step 116 whether the desired file system access is to be made to ROM DISK D:. If not, the system proceeds as in the prior art to access the FAT file system. If the desired access is to be made to ROM DISK D:, it is done through the ROM file system 118 and RAM LOADER 51 by programs IBMDOS.COM 52 and IBMBIO.COM 50. The term "ROM DISK" is defined hereby to mean a ROM unit containing images of files storable on a hard disk or floppy diskette, which images are stored in a packed format in a ROM file system, that is different from the FAT file system used to store files on a disk or diskette.

Feigenbaum, 9:39-60

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Feigenbaum, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Feigenbaum, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> One of the objects of the invention is to provide a ROM based DOS that rapidly tests, boots, and initializes a personal computer and appears to the user to "instantly" start up when the computer is turned on.

Feigenbaum, 3:30-33

> When switch 35 is initially turned on, display 30 warms up within a relatively short period and the ROM based booting and system initialization, described in detail below, occurs within the period required for the display to warm up so that at the end of such period, the display screen becomes visible to the user to give the appearance of an instant startup.

Feigenbaum, 4:63-5:2

> In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user

interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code. Feigenbaum, 10:18-33

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Feigenbaum, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Feigenbaum, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1.3 and 1.4 above.

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

> FIG. 4 illustrates the relative position and use of IFS HANDLER 53. Within the prior art, an application program 110 can be thought of as sitting on top of FAT file system 112, which in turn sits on top of disk/diskette driver 114 which overlies the disk 24 containing the actual files and data. Application 110 uses interrupt INT 21H to access the file system through IBMDOS.COM, which acts through IBMBIO.COM to access the disk. In accordance with the invention, IFS HANDLER 53 is interposed at the file system level in the INT 21H process and decides in step 116 whether the desired file system access is to be made to ROM DISK D:. If not, the system proceeds as in the prior art to access the FAT file system. If the desired access is to be made to ROM DISK D:, it is done through the ROM file system 118 and RAM LOADER 51 by programs IBMDOS.COM 52 and IBMBIO.COM 50. The term "ROM DISK" is defined hereby to mean a ROM unit containing images of files storable on a hard disk or floppy diskette, which images are stored in a packed format in a ROM file system, that is different

from the FAT file system used to store files on a disk or diskette.

Feigenbaum, 9:39-60

> Programs 50 through 56 provide the minimum level of operating system support for operating data processing system 10. IBMBIO.COM 50 (except for RAM LOADER 51), IBMDOS.COM 52 (except for IFS handler 53), and COMMANDS 54 AND 56 (which together form the conventional COMMAND.COM DOS program) comprise the conventional DOS kernel. IBMBIO.COM provides low level device support. IBMDOS.COM provides application support. COMMAND.COM is the command processor. COMMAND 54 stays resident in RAM 14 once it is loaded while COMMAND 56 can be overlaid and later copied into RAM 14 as needed. ROMSHELL 57 provides a graphical user interface and is transient while SHELSTUB 59 is designed to be resident in the RAM for reloading the ROMSHELL program after quitting an application. The ROM DOS 34 programs when loaded into the RAM generally operate the same as the corresponding programs in DISK DOS 36, the only difference being that such programs are loaded from ROM 16 instead of disk 24. Relative to other portions of DOS and the system, ROM DOS 34 is transparent and appears to act the same as corresponding portions of DISK DOS 36. The ROM DOS files are stored in ROM 16 outside of the address space available for DOS programs, and RAM LOADER 51 is able to work outside such address space to copy such programs into the DOS address space in RAM 14. RAM LOADER 51 uses a standard method of moving blocks of data from the protected mode address space into the real mode DOS address space, such as by using the well known system service interrupt 15H of BIOS 40 which is invoked by setting the well known register AH (not shown) of microprocessor 12 to the MOVE BLOCK function 87H. Such function transfers a block of data from storage above the IMB protected mode address range by switching to the protected mode. RAM LOADER 51 is invoked using interrupt 2BH and accesses the information from items 42 and 46 to load the ROM header at initialization time, check for the existence of a module in ROM, load a module from ROM, and access flag 58 in CMOS RAM 20.

Feigenbaum, 6:3-45

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Feigenbaum, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Feigenbaum, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.3, 1.4, and 15 above.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Feigenbaum, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Feigenbaum, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Feigenbaum, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Feigenbaum, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Feigenbaum, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Feigenbaum, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Feigenbaum, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Feigenbaum, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Feigenbaum, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Feigenbaum, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Feigenbaum, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Feigenbaum, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Feigenbaum, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Feigenbaum, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Feigenbaum, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Feigenbaum, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Feigenbaum, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Feigenbaum, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 15 and 27 above	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Feigenbaum, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 16 and 27 above	

Appendix A5
Invalidity of U.S. Patent 7,181,608 based on Greene

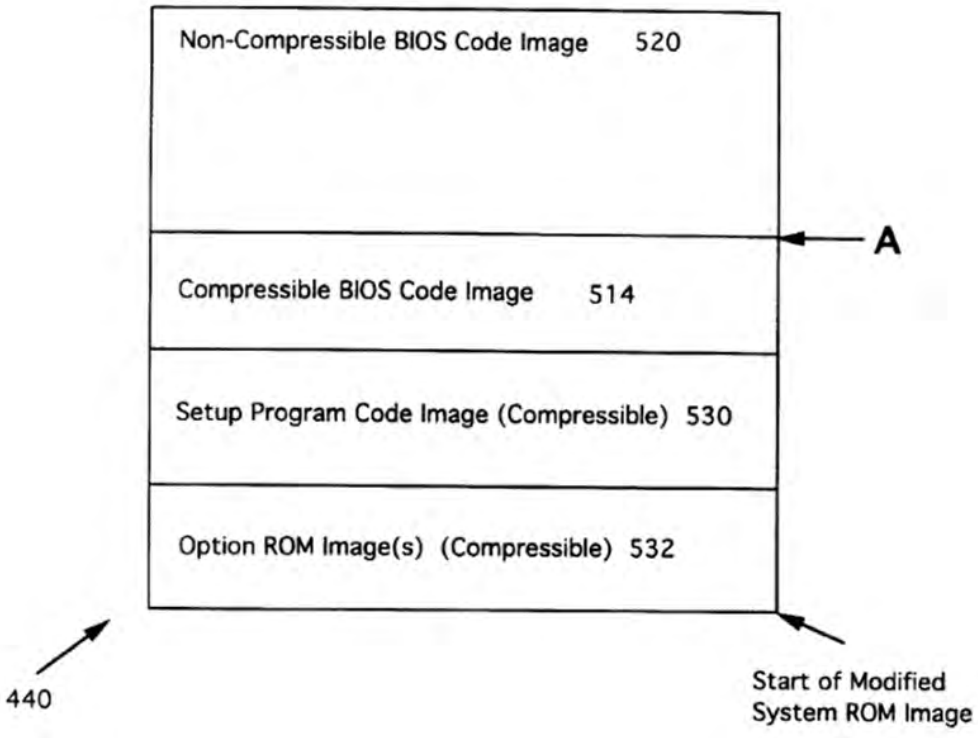
U.S. Patent No. 5,836,013 to Greene ("Greene") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Greene, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:



Greene, Fig. 5.

> "A chipset (platform)-independent method and apparatus for compressing and decompressing a system ROM of a computer (e.g., BIOS, setup program, and one or more option ROMs) are disclosed. The setup program, option ROM, and part of the BIOS are compressed using a lossless compression algorithm. A non-compressible portion of the BIOS includes a decompression algorithm and a shadow RAM block table of chipset-specific registers and bit patterns to write-enable

> and read-enable shadow RAM (RAM that is mapped to the ROM address space). The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data."

Greene, Abstract. *See also* 1:40-1:63.

> "During the BIOS Power-On Self-Test (POST) process, the compressed system ROM is copied to conventional memory, and the decompression program is executed."

Greene, Abstract. *See also* 1:64-2:13.

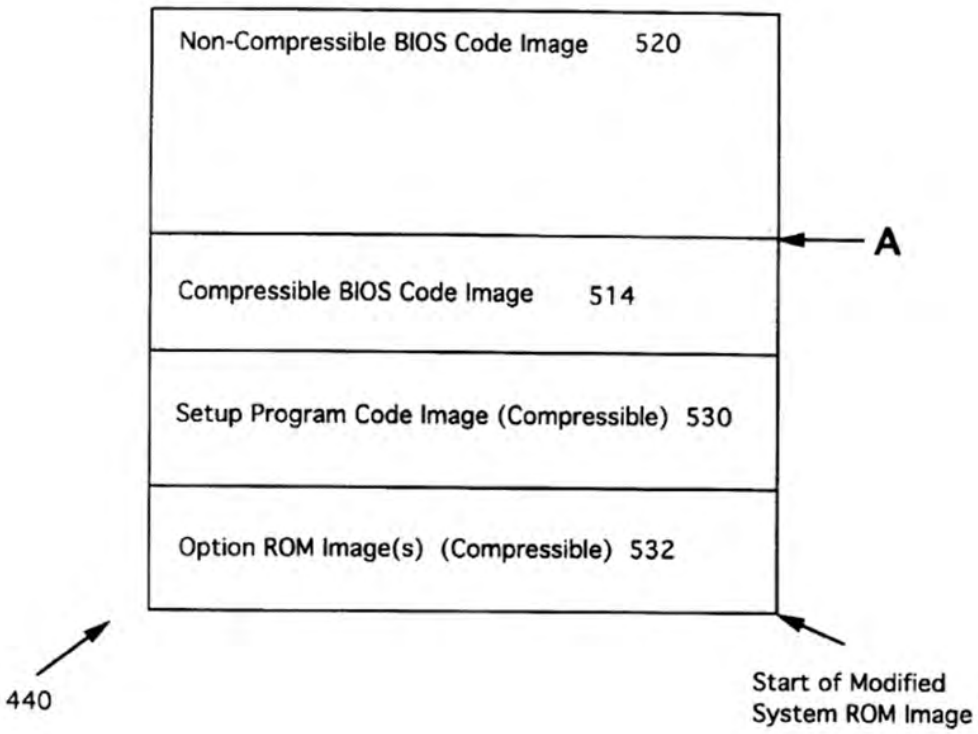
> "System ROM 210 comprises BIOS (Basic Input/Output System) 310, and optionally setup program 320, and one or more option ROM images for input/output (I/O) devices associated with the computer 332. An "image" is a binary representation of code and/or data of a computer file. BIOS 310 comprises a plurality of routines that initialize the computer hardware and provide primitive I/O services that the operating system and application programs use to manipulate hardware associated with the computer. System setup program 320 is a utility that allows the end user to configure BIOS 310."

Greene, 3:16-28, Fig. 3.

1.1 maintaining a list of boot data used for booting a computer system;	Greene, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:



Greene, Fig. 5.

> "A chipset (platform)-independent method and apparatus for compressing and decompressing a system ROM of a computer (e.g., BIOS, setup program, and one or more option ROMs) are disclosed. The setup program, option ROM, and part of the BIOS are compressed using a lossless compression algorithm. A non-compressible portion of the BIOS includes a decompression algorithm and a shadow RAM block table of chipset-specific registers and bit patterns to write-enable and read-enable shadow RAM (RAM that is mapped to the ROM

address space). The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data."

Greene, Abstract. *See also* 1:40-1:63.

"During the BIOS Power-On Self-Test (POST) process, the compressed system ROM is copied to conventional memory, and the decompression program is executed."

Greene, Abstract. *See also* 1:64-2:13.

"System ROM 210 comprises BIOS (Basic Input/Output System) 310, and optionally setup program 320, and one or more option ROM images for input/output (I/O) devices associated with the computer 332. An "image" is a binary representation of code and/or data of a computer file. BIOS 310 comprises a plurality of routines that initialize the computer hardware and provide primitive I/O services that the operating system and application programs use to manipulate hardware associated with the computer. System setup program 320 is a utility that allows the end user to configure BIOS 310."

Greene, 3:16-28, Fig. 3.

"BIOS code 410, and optionally setup program code 430, and one or more option ROM code modules 432, are each developed, written (e.g., in any computer language such as C or Assembly language) and saved in a computer file. Code and data, as referred to herein, are used interchangeably and comprise computer code and/or data. BIOS code 410 is separated 412 into compressible BIOS code 414 and non-compressible BIOS code 420. Non-compressible BIOS code 420 comprises compression-related information table 424, decompression program 422, and shadow RAM block table 1104."

Greene, 3:42-52.

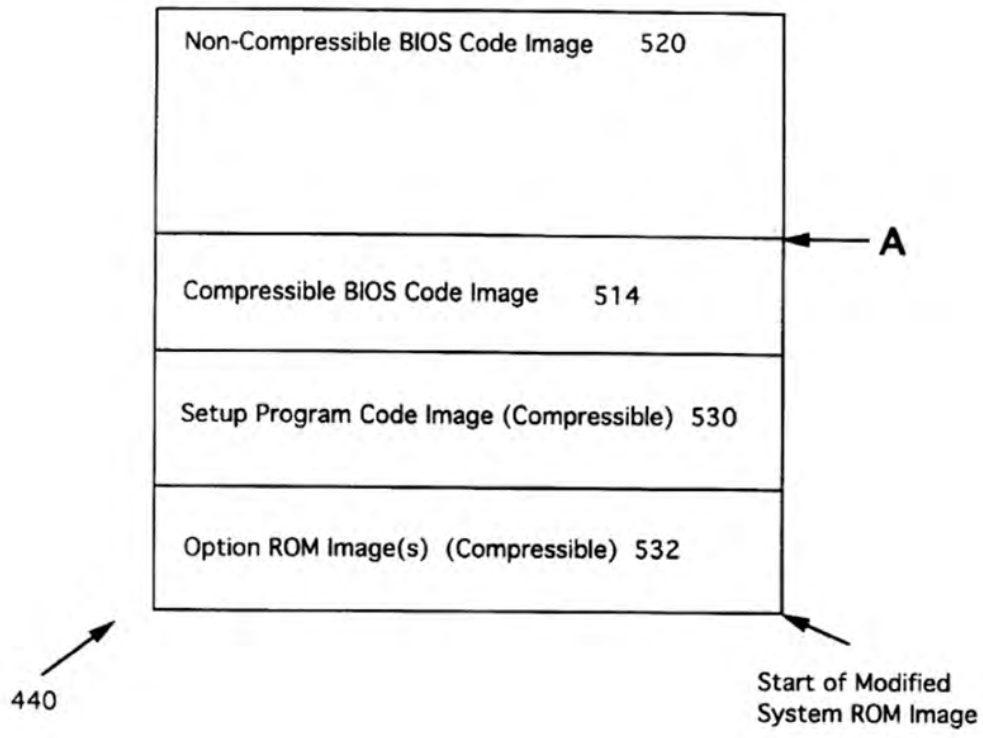
"Continuing with FIG. 4, compressible BIOS code 414, non-compressible BIOS code 420 (comprising decompression program 422 (1102, 1106, 1108), compression-related information table 424, and shadow RAM block table 1104), and optionally setup program code 430, and option ROM code module(s) 432, are compiled or assembled and linked 434 to generate a modified system ROM image 440."

Greene, 4:64-5:3.

1.2 initializing a central processing unit of the computer system;	Greene, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:



Greene, Fig. 5.

> "A chipset (platform)-independent method and apparatus for compressing and decompressing a system ROM of a computer (e.g., BIOS, setup program, and one or more option ROMs) are disclosed. The setup program, option ROM, and part of the BIOS are compressed using a lossless compression algorithm. A non-compressible portion of the BIOS includes a decompression algorithm and a shadow RAM block table of chipset-specific registers and bit patterns to write-enable and read-enable shadow RAM (RAM that is mapped to the ROM

> address space). The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data."

Greene, Abstract. *See also* 1:40-1:63.

> "During the BIOS Power-On Self-Test (POST) process, the compressed system ROM is copied to conventional memory, and the decompression program is executed."

Greene, Abstract. *See also* 1:64-2:13.

> "System ROM 210 comprises BIOS (Basic Input/Output System) 310, and optionally setup program 320, and one or more option ROM images for input/output (I/O) devices associated with the computer 332. An "image" is a binary representation of code and/or data of a computer file. BIOS 310 comprises a plurality of routines that initialize the computer hardware and provide primitive I/O services that the operating system and application programs use to manipulate hardware associated with the computer. System setup program 320 is a utility that allows the end user to configure BIOS 310."

Greene, 3:16-28, Fig. 3.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Greene, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

> "The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data. Thus, the data can be decompressed anywhere in memory of a target computer. For example, the BIOS is decompressed to shadow RAM and the setup program is decompressed to conventional memory."

Greene, Abstract.

> "The decompression program scans the compressed system ROM for compressed data blocks and decompresses the compressed data therein to the associated locations in memory. If compressed data is located in shadow RAM, shadow RAM is enabled for writing and reading with reference to the chipset-specific information in the shadow RAM block table. If compressed data was decompressed to conventional memory space, this space is cleared before exiting the POST process."

Greene, Abstract.

> "Compressed data is stored in a compressed data block comprising the compressed data and various associated information including the location in memory to place the compressed data when decompressed. The compressed system ROM is stored in ROM of a target computer."

Greene, 1:58-63.

> "During the BIOS Power-On Self-Test (POST) process of the target computer, the compressed system ROM image is copied to conventional memory (e.g., RAM), and the decompression program is executed. The decompression program write-enables shadow RAM (with reference to the chipset-specific information in the shadow RAM block table), copies the non-compressible BIOS code image from conventional memory to shadow RAM, and read-enable shadow RAM. The decompression program scans the compressed system ROM image for compressed data blocks and decompresses the compressed data therein to the associated locations in memory."

Greene, 1:64-2:8.

> "In a preferred embodiment, the destination location in memory 100 (e.g., address space) of decompressed data is specified in decompressed data start 722. Thus, compressed data 702 can be placed anywhere in memory 100 upon decompression."

Greene, 5:53-6:2

> "Compressed system ROM image 632 is copied 802 from ROM 130 to conventional memory 110, (e.g., RAM 112)."

Greene, 7:27-29, Fig. 8a.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Greene, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

> "The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data. Thus, the data can be decompressed anywhere in memory of a target computer. For example, the BIOS is decompressed to shadow RAM and the setup program is decompressed to conventional memory."

Greene, Abstract.

> "The decompression program scans the compressed system ROM for compressed data blocks and decompresses the compressed data therein to the associated locations in memory. If compressed data is located in shadow RAM, shadow RAM is enabled for writing and reading with reference to the chipset-specific information in the shadow RAM block table. If compressed data was decompressed to conventional memory space, this space is cleared before exiting the POST process."

Greene, Abstract.

> "During the BIOS Power-On Self-Test (POST) process of the target

> computer, the compressed system ROM image is copied to conventional memory (e.g., RAM), and the decompression program is executed. The decompression program write-enables shadow RAM (with reference to the the chipset-specific information in the shadow RAM block table), copies the non-compressible BIOS code image from conventional memory to shadow RAM, and read-enables shadow RAM. The decompression program scans the compressed system ROM image for compressed data blocks and decompresses the compressed data therein to the associated locations in memory. If data is located in shadow RAM, shadow RAM is write-enabled and read-enabled (with reference to the chipset-specific information in the shadow RAM block table). If compressed data was decompressed to conventional memory, this space is cleared before exiting the POST process."

Greene, 1:64-2:13.

> "Referring to FIG. 10, decompression program 422 comprises a decompression algorithm 1102. Decompression algorithm 1102 corresponds to the compression algorithm used (e.g., LZSS or LZARI)."

Greene, 3:66-4:2.

> "Compressed system ROM 632 image must be decompressed before the compressed data therein can be used on the target computer. In a preferred embodiment, decompression is done early in the POST process."

Greene, 7:21-24.

> "Program execution control is transferred 804 to decompression program 422 (in non-compressible BIOS code image 520 of compressed system ROM image 632)."

Greene, 7:29-32, Fig. 8a. *See also* 7:33-7:64.

> "The decompression scan process repeats 832 until each compressed data block 700 in compressed system ROM image 632 is decompressed. In a preferred embodiment, program execution control is then transferred 834 to BIOS 310, now running in shadow RAM 202. In one embodiment, if data was decompressed to conventional (RAM) memory 110, conventional memory 110 is cleared 836 at the end of the POST process and before the operating system is booted (for compatibility reasons)."

Greene, 7:65-8:21.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Greene, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "BIOS 310 comprises a plurality of routines that initialize the computer hardware and provide primitive I/O services that the operating system and application programs use to manipulate hardware associated with the computer."

Greene, 3:21-25

> "Conventional memory 110 comprises a random access memory (RAM) 112 address space that is set aside for use by operating systems and application programs."

Greene, 2:50-53.

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Greene, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.3, and 1.4 above.	

4. The method of claim 1, further comprising updating the list of boot data.	Greene, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.1 above.	

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Greene, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Greene, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Greene, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Greene, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.2, 1.3, and 1.4 above.	

7.2 a cache memory device; and;	Greene, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Greene, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Greene, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Greene, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Greene, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> "BIOS 310 comprises a plurality of routines that initialize the computer hardware and provide primitive I/O services that the operating system and application programs use to manipulate hardware associated with the computer."

Greene, 3:21-25

> "Conventional memory 110 comprises a random access memory (RAM) 112 address space that is set aside for use by operating systems and application programs."

Greene, 2:50-53.

9.1 maintaining a list of application data associated with an application program;	Greene, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Greene, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Greene, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> "BIOS 310 comprises a plurality of routines that initialize the computer hardware and provide primitive I/O services that the operating system and application programs use to manipulate hardware associated with the computer."

Greene, 3:21-25

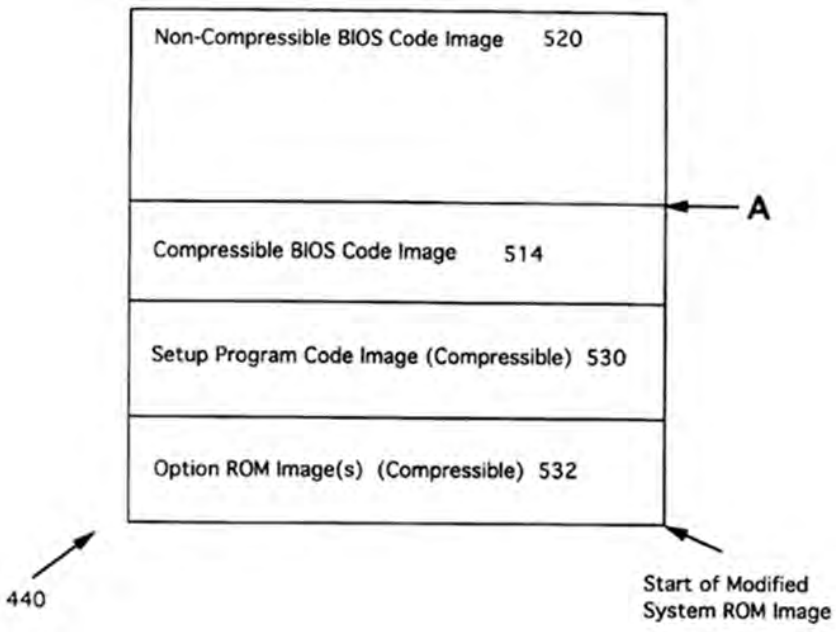
> "Conventional memory 110 comprises a random access memory (RAM) 112 address space that is set aside for use by operating systems and application programs."

Greene, 2:50-53.

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Greene, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:



Greene, Fig. 5.

> "A chipset (platform)-independent method and apparatus for compressing and decompressing a system ROM of a computer (e.g., BIOS, setup program, and one or more option ROMs) are disclosed. The setup program, option ROM, and part of the BIOS are compressed using a lossless compression algorithm. A non-compressible portion of the BIOS includes a decompression algorithm and a shadow RAM

> block table of chipset-specific registers and bit patterns to write-enable and read-enable shadow RAM (RAM that is mapped to the ROM address space). The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data."

Greene, Abstract. *See also* 1:40-1:63.

> "During the BIOS Power-On Self-Test (POST) process of the target computer, the compressed system ROM image is copied to conventional memory (e.g., RAM), and the decompression program is executed. The decompression program write-enables shadow RAM (with reference to the chipset-specific information in the shadow RAM block table), copies the non-compressible BIOS code image from conventional memory to shadow RAM, and read-enable shadow RAM. The decompression program scans the compressed system ROM image for compressed data blocks and decompresses the compressed data therein to the associated locations in memory."

Greene, 1:64-2:8.

> "The setup program code, option ROM code, compressible BIOS code, and non-compressible BIOS code are compiled or assembled and linked to create a modified system ROM image. The starting address of the non-compressible BIOS code image in the modified system ROM image is stored. A lossless compression algorithm is used to compress the modified system ROM image (up to the non-compressible BIOS code image address), thereby generating a compressed system ROM image. Compressed data is stored in a compressed data block comprising the compressed data and various associated information including the location in memory to place the compressed data when decompressed. The compressed system ROM is stored in ROM of a target computer."

Greene, 1:50-63.

> "BIOS code 410, and optionally setup program code 430, and one or more option ROM code modules 432, are each developed, written (e.g., in any computer language such as C or Assembly language) and saved in a computer file. Code and data, as referred to herein, are used interchangeably and comprise computer code and/or data. BIOS code 410 is separated 412 into compressible BIOS code 414 and non-compressible BIOS code 420. Non-compressible BIOS code 420 comprises compression-related information table 424, decompression

program 422, and shadow RAM block table 1104."

Greene, 3:42-52.

> "Continuing with FIG. 4, compressible BIOS code 414, non-compressible BIOS code 420 (comprising decompression program 422 (1102, 1106, 1108), compression-related information table 424, and shadow RAM block table 1104), and optionally setup program code 430, and option ROM code module(s) 432, are compiled or assembled and linked 434 to generate a modified system ROM image 440."

Greene, 4:64-5:3.

> "Compression-related information table 424, comprises, for example, a preferred compression algorithm to use when compressing system ROM image 440 (see below step 630). In a preferred embodiment, compression algorithm is a lossless decompression algorithm, for example, LZSS or LZARI, which are commercially available compression/ decompression algorithms."

Greene, 3:53-59.

> "Referring now to FIG. 6, modified system ROM image 440 is input to compression utility 600. Compression utility 600 uses compression-related information table 424 (in modified system ROM image 440) to determine 610 a compression algorithm 612 to use in compressing modified system ROM image 440. As discussed above, compression algorithm 612 can be any lossless compression algorithm, but must correspond to the decompression algorithm (1102 below) used in the system ROM."

Greene, 5:19-27, Fig. 6.

> "Knowing the location of non-compressible BIOS code image 520 in modified system ROM image 440, compression utility 600 uses compression algorithm 612 to compress 630 each compressible image (514, 530, 532) in modified system ROM image 440. Each compressed image is stored in a compressed data block 700."

Greene, 5:31-36.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Greene, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

> "During the BIOS Power-On Self-Test (POST) process of the target computer, the compressed system ROM image is copied to conventional memory (e.g., RAM), and the decompression program is executed. The decompression program write-enables shadow RAM (with reference to the the chipset-specific information in the shadow RAM block table), copies the non-compressible BIOS code image from conventional memory to shadow RAM, and read-enables shadow RAM. The decompression program scans the compressed system ROM image for compressed data blocks and decompresses the compressed data therein to the associated locations in memory. If data is located in shadow RAM, shadow RAM is write-enabled and read-enabled (with reference to the chipset-specific information in the shadow RAM block table). If compressed data was decompressed to conventional memory, this space is cleared before exiting the POST process."

Greene, 1:64-2:13.

> "Referring to FIG. 10, decompression program 422 comprises a decompression algorithm 1102. Decompression algorithm 1102 corresponds to the compression algorithm used (e.g., LZSS or LZARI)."

Greene, 3:66-4:2.

> "Compressed system ROM 632 image must be decompressed before the compressed data therein can be used on the target computer. In a preferred embodiment, decompression is done early in the POST process."

Greene, 7:21-24.

> "Program execution control is transferred 804 to decompression program 422 (in non-compressible BIOS code image 520 of compressed system

ROM image 632)."

Greene, 7:29-32, Fig. 8a. *See also* 7:33-7:64.

"The decompression scan process repeats 832 until each compressed data block 700 in compressed system ROM image 632 is decompressed. In a preferred embodiment, program execution control is then transferred 834 to BIOS 310, now running in shadow RAM 202. In one embodiment, if data was decompressed to conventional (RAM) memory 110, conventional memory 110 is cleared 836 at the end of the POST process and before the operating system is booted (for compatibility reasons)."

Greene, 7:65-8:21.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Greene, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Greene, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Greene, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See Claims 1.3 and 1.4 above.

See also

> "A lossless compression algorithm is used to compress the modified system ROM image (up to the non-compressible BIOS code image address), thereby generating a compressed system ROM image."

Greene, 1:54-58.

> "In a preferred embodiment, compression algorithm is a lossless decompression algorithm, for example, LZSS or LZARI, which are commercially available compression/ decompression algorithms. In general, LZSS decompresses faster than LZARI. LZARI generally compresses data better (smaller) than LZSS. For performance reasons, it is suggested that LZARI decompression code be used with an Intel 80486 or better CPU."

Greene, 3:56-63.

> "Referring to FIG. 10, decompression program 422 comprises a decompression algorithm 1102. Decompression algorithm 1102 corresponds to the compression algorithm used (e.g., LZSS or LZARI)."

Greene, 3:66-4:2.

> "As discussed above, compression algorithm 612 can be any lossless compression algorithm, but must correspond to the decompression algorithm (1102 below) used in the system ROM."

Greene, 5:24-27.

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Greene, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

See also

> "A lossless compression algorithm is used to compress the modified system ROM image (up to the non-compressible BIOS code image address), thereby generating a compressed system ROM image."

Greene, 1:54-58.

> "Referring now to FIG. 6, modified system ROM image 440 is input to compression utility 600. Compression utility 600 uses compression-related information table 424 (in modified system ROM image 440) to determine 610 a compression algorithm 612 to use in compressing modified system ROM image 440. As discussed above, compression algorithm 612 can be any lossless compression algorithm, but must correspond to the decompression algorithm (1102 below) used in the system ROM."

Greene, 5:19-27, Fig. 6.

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Greene, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Greene, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above.	

22.1 maintaining a list of boot data used for booting a computer system;.	Greene, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.1 above.	

22.2 initializing a central processing unit of the computer system;.	Greene, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.2 above.	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Greene, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.3 above.	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Greene, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.4 above.	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Greene, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 4, 10 and 11 above.	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Greene, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 15 and 22 above.	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Greene, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 16 and 22 above.	

27.1 a boot device..	Greene, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1 above.	

27.2 a processor..	Greene, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.2 above.	

27.3 cache memory; and.	Greene, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.3 and 1.4 above.	

27.4 non-volatile memory for storing logic code for use by the processor,..	Greene, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.1 and 1.3 above.	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Greene, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.1 above.	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Greene, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.3 above.	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Greene, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.4 above.	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Greene, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 4, 10, and 11 above.	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Greene, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 15 and 27 above.	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Greene, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 16 and 27 above.	

Appendix A6
Invalidity of U.S. Patent 7,181,608 based on Hillis

U.S. Patent No. 6,421,776 to Hillis ("Hillis") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A6
Invalidity of U.S. Patent 7,181,608 based on Hillis

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Hillis, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: "Another advantage is in reducing the time required for bootstrapping." Hillis, 3:20-21. "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory." Hillis, 7:66-8:3.	

1.1 maintaining a list of boot data used for booting a computer system;	Hillis, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

> "After locating a disk with a valid boot record, the BIOS program reads the data stored on the first sector of the disk, and copies that data to specific locations in RAM. This information, found in the same location on every formatted disk, constitutes the DOS boot record. The BIOS then passes control to the boot record which instructs the PC on how to load the two hidden operating system files to RAM (the files named IBMBIO.COM and IBMDOS.COM on IBM computers). After loading other operating system files into RAM to carry out the rest of the boot up sequence, the boot record is no longer needed."

Hillis, 1:46-56.

1.2 initializing a central processing unit of the computer system;	Hillis, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

> "Upon system initialization during a cold boot, the uncompressed portion of POST is executed from the ROM to enable the system memory, and then an image of the BIOS code is written to shadow memory."

Hillis, Abstract.

> "In accordance with an aspect of the invention, the portion of the BIOS code that is uncompressed in ROM includes an initial portion of a power on system test (POST) code which is sufficient to enable the system memory, a remaining portion of which is compressed."

Hillis, 3:44-48.

> "Upon cold boot, the initial portion POST is read directly from ROM to enable the system memory, and then an image of the entire BIOS code, the major portion of which is in compressed form, is written to RAM in the system memory, and control is transferred to the image."

Hillis, 3:54-58.

> "The phase 1 POST code, which is stored in uncompressed (unpacked) form, consists of only that portion of POST that is necessary to enable the system memory. That is, under conventional BIOS protocol, the initial portion of POST is first read from the BIOS ROM to enable or "wake up" the system memory, usually composed of CMOS type random access semiconductor memory."

Hillis, 3:42-48.

> "As shall be described in more detail hereinafter, upon system initialization (bootstrapping), an image of the BIOS ROM is copied to the

upper 128 kbyte region of system memory, or shadow RAM, from Where system execution takes place for higher operating speed."

Hillis: 6:1-4.

See also Hillis, 6:16-49.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Hillis, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

> "To increase the effective capacity of BIOS, an initial portion of the power on system reset (POST) code that is required to enable the system memory is stored in ROM in uncompressed form, and substantially the remaining portion of the BIOS code is stored in compressed form."

Hillis, Abstract.

> "Upon system initialization during a cold boot, the uncompressed portion of POST is executed from the ROM to enable the system memory, and then an image of the BIOS code is written to shadow memory."

Hillis, Abstract.

> "In accordance with an important aspect of the invention, an initial portion of the BIOS code that is required to enable the system memory is in uncompressed form and a remaining portion thereof for carrying out prescribed functions including converting operating signals developed by an operating system executed by the CPU into electrical signals compatible with devices that are responsive to signals provided by the CPU to the system bus, is in compressed form."

Hillis, 3:36-43.

> "To reduce the time required for decompression of BIOS code, the code

is transferred from ROM to the system memory in compressed form."

Hillis, 3:49-51.

"Upon cold boot, the initial portion POST is read directly from ROM to enable the system memory, and then an image of the entire BIOS code, the major portion of which is in compressed form, is written to RAM in the system memory, and control is transferred to the image."

Hillis, 3:54-58.

"The next layers of the BIOS ROM consist of compressed set-up data, including descriptive text, and compressed setup code. Next, all but the initial portion of the power on system test (POST) code (termed "phase 2 POST herein") is stored in compressed form, the initial portion of POST (termed "phase 1 POST") being stored in the next lower layer of BIOS ROM."

Hillis, 5:35-41.

"Then, all the remaining portion of POST and other BIOS code are copied to memory in a region thereof termed "shadow RAM" or "shadow memory."

Hillis, 5:48-55.

"As shall be described in more detail hereinafter, upon system initialization (bootstrapping), an image of the BIOS ROM is copied to the upper 128 kbyte region of system memory, or shadow RAM, from where system execution takes place for higher operating speed. Shadowing of the BIOS is well known. In accordance with the invention, however, most of the BIOS code is stored in ROM in compressed form."

Hillis, 6:1-8.

"Next, for improved performance the entire ROM image is transferred to the system memory (step 50) and the microprocessor cache is turned on (step 52). Control of the system is now transferred to the RAM image of POST, shown in FIG. 4 (step 54)."

Hillis, 6:50-54.

"Next, the uncompressed images are copied from the current region of system RAM back into the shadow RAM (step 58)"

Hillis, 6:61-63.

> "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory."

Hillis, 7:66-8:3.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Hillis, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

> "As BIOS code is needed during the remainder of the boot, the code is selectively decompressed from the shadow memory to another region of the system memory to which control is transferred."

Hillis, Abstract.

> "A further advantage is in performing BIOS code decompression under different boot sceneries, cold and warm, and upon memory conditions of real and protect."

Hillis, 3:21-24.

> "Then, after a jump from one location of the system memory to another, decompression of the code takes place."

Hillis, 3:51-53.

> "As needed, portions of the BIOS code including POST, Setup (if invoked) and then other BIOS routines are selectively decompressed

> from the shadow memory to another location of the system memory. Normal execution of POST and BIOS then proceeds until the boot is completed."

Hillis, 3:58-63.

> "Mapping from the BIOS ROM to the system memory is followed by decompression only of those BIOS routines that are necessary. Referring to FIG. 5 depicting the BIOS decompressed shadow RAM image, the upper 64K of shadow memory contains decompressed BIOS code, the lowest 32K portion of the upper 128 kbyte block contains decompressed phase 2 POST code, decompressed set-up data and code reside at the lowest 128K in system memory, and decompressed video and SES reside as shown."

Hillis, 6:8-16.

> "As BIOS routines are needed, they are now selectively decompressed from system RAM to system RAM (step 56). The POST may be decompressed into any other region of RAM within or outside the system RAM range of addresses including the region ultimately to be occupied by the operating system."

Hillis, 6:55-60.

> "Decompression occurs selectively in a similar manner for other compressed images of the BIOS code, other than setup which in this example has not yet been called (step 60). Control of the system is transferred to the shadow image of POST (step 62). Execution of POST continues."

Hillis, 6:63-7:1.

> "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory."

Hillis, 7:66-8:3.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Hillis, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "Next loaded from the boot disk into RAM is the file COMMAND.COM which is an operating system file containing, among other functions, fundamental DOS commands used throughout application program execution, and a file named AUTOEXECBAT created by the user and containing a series of DOS batch file commands or program names to be executed by the PC each time the computer is turned on."

Hillis, 1:65-2:5.

> "This is followed in the same region of memory by the operating system, such as DOS, followed by any application programs."

Hillis, 2:14-16.

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Hillis, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

> "The computer 10 which preferably, but not necessarily, is of a type utilizing an IBM Personal Computer or similar system, includes a console housing 12 within which circuit boards containing the necessary circuitry including microprocessor and BIOS chips, controllers, random access memory and other hardware are arranged."

Hillis, 5:6-11.

> "Data are stored in floppy, CD-ROM and hard disk drives 28, 32 and 34 for access by the CPU 24 through corresponding controllers 30. Display 14 is connected to the system bus 22 through a video controller 36."

Hillis, 5:20-25.

4. The method of claim 1, further comprising updating the list of boot data.	Hillis, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.1 above.	

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Hillis, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Hillis, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.1 and 4 above.	

Appendix A6
Invalidity of U.S. Patent 7,181,608 based on Hillis

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Hillis, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Hillis, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

> "The computer 10 which preferably, but not necessarily, is of a type utilizing an IBM Personal Computer or similar system, includes a console housing 12 within which circuit boards containing the necessary circuitry including microprocessor and BIOS chips, controllers, random access memory and other hardware are arranged."

Hillis, 5:6-11.

> "Data are stored in floppy, CD-ROM and hard disk drives 28, 32 and 34 for access by the CPU 24 through corresponding controllers 30. Display 14 is connected to the system bus 22 through a video controller 36."

Hillis, 5:20-25.

7.2 a cache memory device; and;	Hillis, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Hillis, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Hillis, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Hillis, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Hillis, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> "Next loaded from the boot disk into RAM is the file COMMAND.COM which is an operating system file containing, among other functions, fundamental DOS commands used throughout application program execution, and a file named AUTOEXECBAT created by the user and containing a series of DOS batch file commands or program names to be executed by the PC each time the computer is turned on."

Hillis, 1:65-2:5.

> "This is followed in the same region of memory by the operating system, such as DOS, followed by any application programs."

Appendix A6
Invalidity of U.S. Patent 7,181,608 based on Hillis

Hillis, 2:14-16.

9.1 maintaining a list of application data associated with an application program;	Hillis, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Hillis, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Hillis, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> "Next loaded from the boot disk into RAM is the file COMMAND.COM which is an operating system file containing, among other functions, fundamental DOS commands used throughout application program execution, and a file named AUTOEXECBAT created by the user and containing a series of DOS batch file commands or program names to be executed by the PC each time the computer is turned on."

Hillis, 1:65-2:5.

> "This is followed in the same region of memory by the operating system, such as DOS, followed by any application programs."

Hillis, 2:14-16.

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Hillis, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

> "To increase the effective capacity of BIOS, an initial portion of the power on system reset (POST) code that is required to enable the system memory is stored in ROM in uncompressed form, and substantially the remaining portion of the BIOS code is stored in compressed form."

Hillis, Abstract.

> "After locating a disk with a valid boot record, the BIOS program reads the data stored on the first sector of the disk, and copies that data to specific locations in RAM. This information, found in the same location on every formatted disk, constitutes the DOS boot record. The BIOS then passes control to the boot record which instructs the PC on how to load the two hidden operating system files to RAM (the files named IBMBIO.COM and IBMDOS.COM on IBM computers). After loading other operating system files into RAM to carry out the rest of the boot up sequence, the boot record is no longer needed."

Hillis, 1:46-56.

> "In accordance with an important aspect of the invention, an initial portion of the BIOS code that is required to enable the system memory is in uncompressed form and a remaining portion thereof for carrying out prescribed functions including converting operating signals developed by an operating system executed by the CPU into electrical signals compatible with devices that are responsive to signals provided by the CPU to the system bus, is in compressed form."

Hillis, 3:36-43.

> "The next layers of the BIOS ROM consist of compressed set-up data, including descriptive text, and compressed setup code. Next, all but the initial portion of the power on system test (POST) code (termed "phase 2 POST herein") is stored in compressed form, the initial portion of POST (termed "phase 1 POST") being stored in the next lower layer of BIOS ROM."

Hillis, 5:35-41.

> "As shall be described in more detail hereinafter, upon system initialization (bootstrapping), an image of the BIOS ROM is copied to the upper 128 kbyte region of system memory, or shadow RAM, from where system execution takes place for higher operating speed. Shadowing of the BIOS is well known. In accordance with the invention, however, most of the BIOS code is stored in ROM in compressed form."

Hillis, 6:1-8.

> "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory."

Hillis, 7:66-8:3.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Hillis, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

> "A further advantage is in performing BIOS code decompression under different boot sceneries, cold and warm, and upon memory conditions of real and protect."

Hillis, 3:21-24.

> "As needed, portions of the BIOS code including POST, Setup (if invoked) and then other BIOS routines are selectively decompressed from the shadow memory to another location of the system memory. Normal execution of POST and BIOS then proceeds until the boot is completed."

Hillis, 3:58-63.

> "As BIOS routines are needed, they are now selectively decompressed from system RAM to system RAM (step 56). The POST may be decompressed into any other region of RAM within or outside the system RAM range of addresses including the region ultimately to be occupied by the operating system."

Hillis, 6:55-60.

> "Decompression occurs selectively in a similar manner for other compressed images of the BIOS code, other than setup which in this example has not yet been called (step 60). Control of the system is transferred to the shadow image of POST (step 62)."

Hillis, 6:63-67.

> "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST

code in BIOS ROM, sufficient to enable the system memory." Hillis, 7:66-8:3.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Hillis, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Hillis, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Hillis, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claims 1.3 and 1.4 above.

See also

> "Compression of set-up data and code, phase 2 POST, video, Smart Energy System (TM) and BIOS code is carried out by any commercially available LZ-1 or LZ-2 algorithm, such as the techniques identified in US. Pat. Nos. 4,701,745 and 5,016,009, incorporated herein by reference. Other suitable compression and decompression algorithms, can be used, however."

Hillis, 5:61-67.

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Hillis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

See also

> "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory."

Hillis, 7:66-8:3.

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Hillis, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Hillis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above.	

22.1 maintaining a list of boot data used for booting a computer system;.	Hillis, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.1 above.	

22.2 initializing a central processing unit of the computer system;.	Hillis, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.2 above.	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Hillis, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.3 above.	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Hillis, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.4 above.	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Hillis, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 4, 10 and 11 above.	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Hillis, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 15 and 22 above.	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Hillis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 16 and 22 above.	

27.1 a boot device..	Hillis, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1 above.	

27.2 a processor..	Hillis, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.2 above.	

27.3 cache memory; and.	Hillis, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.3 and 1.4 above.	

27.4 non-volatile memory for storing logic code for use by the processor,..	Hillis, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.1 and 1.3 above.	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Hillis, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.1 above.	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Hillis, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.3 above.	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Hillis, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.4 above.	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Hillis, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 4, 10, and 11 above.	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Hillis, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 15 and 27 above.	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Hillis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 16 and 27 above.	

Appendix A7
Invalidity of U.S. Patent 7,181,608 based on Horning

U.S. Patent No. 5,420,998 to Horning ("Horning") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

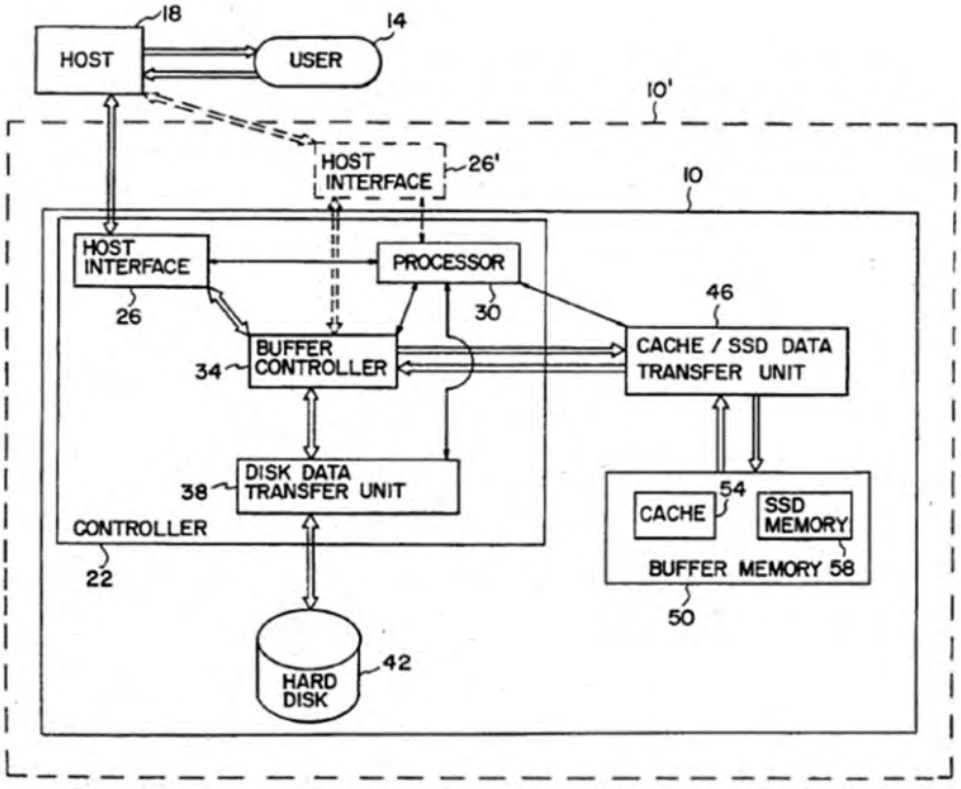
The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A7
Invalidity of U.S. Patent 7,181,608 based on Horning

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Horning, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:



Horning, Fig. 1. *See also* Horning, 2:42-4:15, 4:42-68.

> "Various strategies have been devised to increase the data transfer rate of hard disk technology. One such strategy combines a relatively small volatile solid state data cache together with a hard disk such that data items that are most frequently accessed are maintained in the cache. Since such a cache has a substantially higher data transfer rate than a

> hard disk, a significant increase in host data processing efficiency can be obtained."

Horning, 1:19-27.

> "In summary, by coalescing a hard disk drive and a solid state disk drive into a single unit, this invention provides a large non-volatile data store and a smaller data store with extremely fast data transfers for data known to be accessed frequently."

Horning, 3:59-63.

> "The hard disk 42 data source provides persistent or nonvolatile data storage together with the necessary data accessing mechanisms and logic for reading and writing data. The buffer memory 50 is preferably solid state memory providing data storage plus data transfer rates much higher than the hard disk 42, in fact, preferably rates greater than the transfer rate of the host 18. Thus, both the cache memory 54 and the SSD memory 58 are intended to supply storage for those data items that are accessed frequently by the host 18. The cache memory 54 stores frequently accessed copies of hard disk 42 data items, preferably without any control bits relating to hard disk 42 data formatting. The SSD memory 58 stores frequently accessed data items in a "disk-like" format."

Horning, 5:17-31.

1.1 maintaining a list of boot data used for booting a computer system;	Horning, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

> "The hard disk 42 data source provides persistent or nonvolatile data storage together with the necessary data accessing mechanisms and logic for reading and writing data. The buffer memory 50 is preferably solid state memory providing data storage plus data transfer rates much higher than the hard disk 42, in fact, preferably rates greater than the transfer rate of the host 18. Thus, both the cache memory 54 and the SSD memory 58 are intended to supply storage for those data items that are accessed frequently by the host 18. The cache memory 54 stores frequently accessed copies of hard disk 42 data items, preferably without any control bits relating to hard disk 42 data formatting. The SSD memory 58 stores frequently accessed data items in a "disk-like" format."

Horning, 5:17-31.

> "Depending on the dual disk drive 10 configuration, the microcode instructions for the processor 30 can reside in either a read only memory (ROM) associated with processor 30 or, alternatively, the microcode can reside on the hard disk 42."

Horning, 7:46-50, Fig. 2.

> "In step 68, the processor 30 requests the disk data transfer unit 38 to retrieve the SSD memory 58, or more generally the buffer memory 50, configuration parameter values from a manufacturer specified location on the hard disk 42."

Horning, 7:59-63. *See also* Horning, 7:59-8:2.

> "The configuring is similar to step 220 in Fig. 4 in that it includes providing the cache/SSD data transfer unit 46 with: (i) the number of blocks to be read, "blk_cnt," (ii) the initial header location address, in the

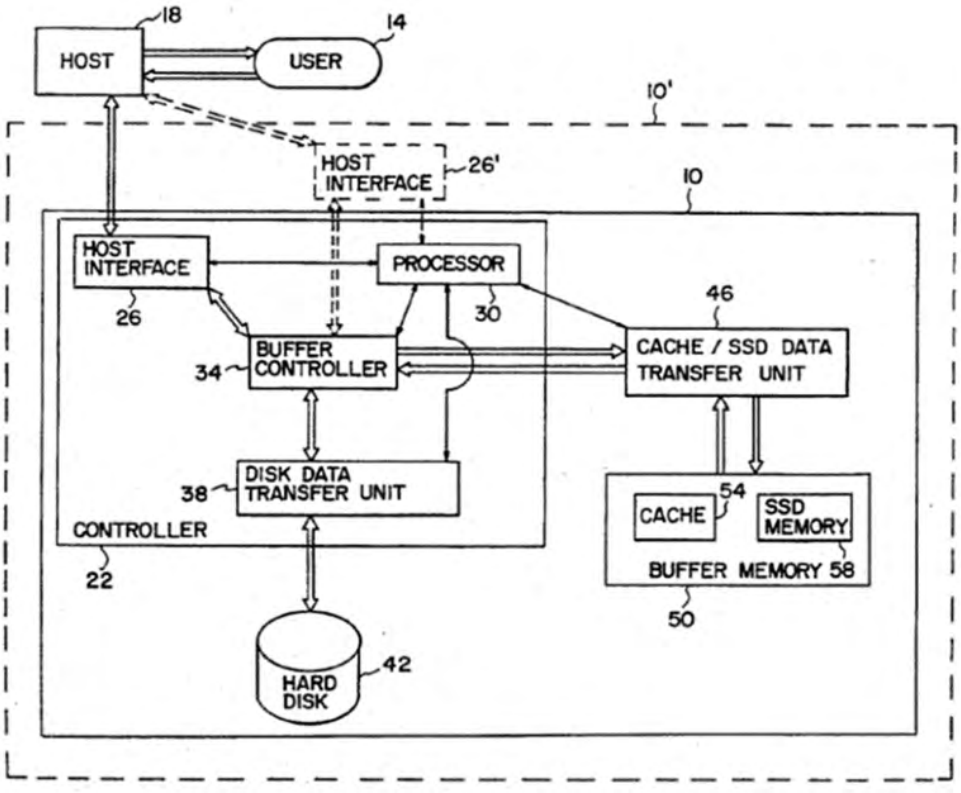
SSD memory 58 where the data is to be read as determined from the physical address established in step 312 above, (iii) the expected value of the sector header at this location, and (iv) a signal indicating that the data to be received from the SSD memory 58 is to be transferred to the host 18 without header and error correction bits."

Horning, 13:2-12.

1.2 initializing a central processing unit of the computer system;	Horning, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:



Horning, Fig. 1.

> "Additional software is also provided to initialize the dual disk drive and to assure SSD data integrity upon power failure. To make the initialization of the dual disk drive as simple as possible for system administration personnel, the initialization procedure has been constructed so that the dual disk drive can be initialized comparable to a conventional hard disk drive. In this case, the parameters for configuring the dual disk drive are supplied with default values during dual disk drive

initialization."

Horning, 3:19-28. *See also*, Horning, 3:28-35.

> "Returning to the controller 22, it includes: a host interface 26, a buffer controller 34, a disk data transfer unit 38 and a processor 30."

Horning, 5:49-52.

> "Referring now to FIG. 2, a preferred procedure is presented for initialization of the dual disk drive 10. In step 60 a user physically switches on power to the dual disk drive 10. Assuming the data connection between the dual disk drive 10 and the host 18 has been physically established, the user preferably activates all further initialization tasks from the host 18. In step 64, both the hard disk 42 and the processor 30 become fully functional. That is, the hard disk 42 is directed to spin-up (i.e. accelerate rotation of its included magnetic storage disk until the proper rotation rate is attained to allow data to be transferred) and the processor 30 is directed to boot-up."

Horning, 7:34-46, Fig. 2.

> "If the microcode resides in ROM, then the processor 30 can boot-up simultaneously with the hard disk 42 spin-up. Otherwise, the processor 30 boot-up will occur immediately after spin-up and the microcode is downloaded into processor 30."

Horning, 7:50-55.

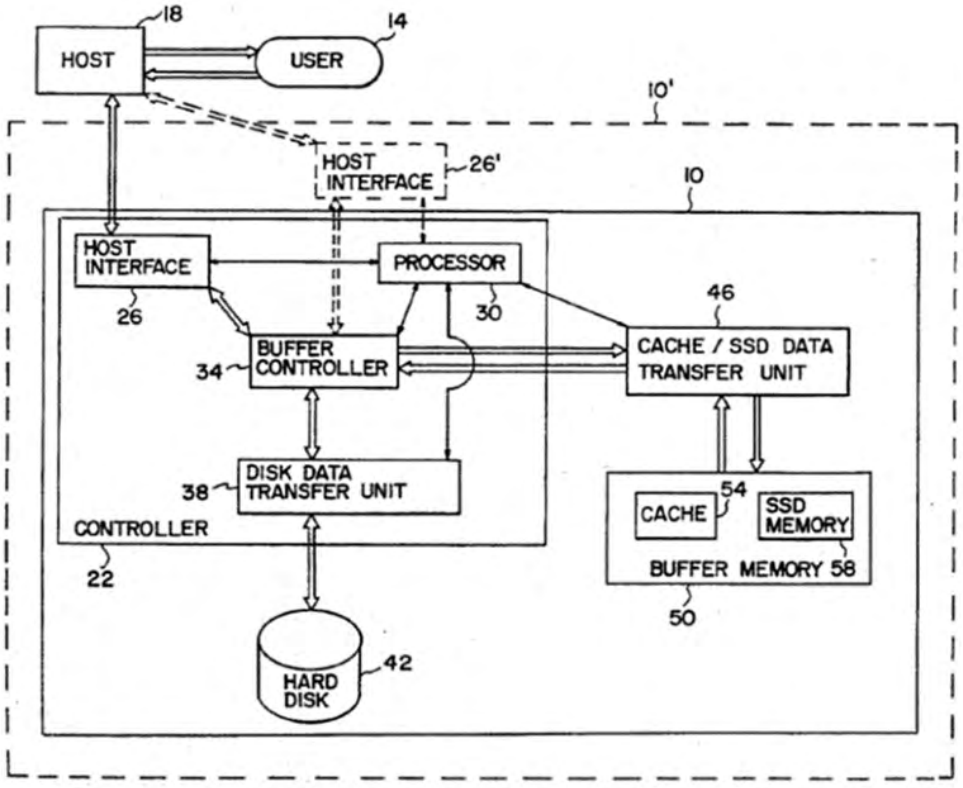
> "Upon completion of the SSD memory 58 formatting, the cache/SSD data transfer unit 46 interrupts the processor 30 signaling that SSD memory 58 initialization is complete."

Horning, 8:16-19.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Horning, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:



Horning, Fig. 1.

> "The solid state buffer memory, normally used as a data cache in a conventional hard disk drive, in accordance with the present invention, is partitioned into two portions one portion remains as the hard disk data cache as in the conventional hard disk drive configuration while the other portion is used as a memory for an SSD such that the SSD memory is considered by the host to be an entirely separate storage device from that of the hard disk."

Horning, 2:46-55.

> "The hard disk 42 data source provides persistent or nonvolatile data storage together with the necessary data accessing mechanisms and logic for reading and writing data. The buffer memory 50 is preferably solid state memory providing data storage plus data transfer rates much higher than the hard disk 42, in fact, preferably rates greater than the transfer rate of the host 18. Thus, both the cache memory 54 and the SSD memory 58 are intended to supply storage for those data items that are accessed frequently by the host 18. The cache memory 54 stores frequently accessed copies of hard disk 42 data items, preferably without any control bits relating to hard disk 42 data formatting. The SSD memory 58 stores frequently accessed data items in a "disk-like" format."

Horning, 5:17-31.

> "If the data transfer is directed to a hard disk 42 location, then preferably the transfer must be through the cache memory 54 regardless of whether the host request is a read or write. That is, if a data read is requested then a- valid copy of the requested data must either reside in the cache memory 54 or be transferred into the cache memory 54 from the hard disk 42 via the disk data transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 46. In any case, the requested data is subsequently transferred from the cache memory 54 to the host 18, via the cache/SSD data transfer unit 46, the buffer controller 34 and the host interface 26."

Horning, 6:33-45.

> "Of course, there are well known caching strategies that can be employed to facilitate these data transfers."

Horning, 6:52-54. *See also* Horning, 6:54-7:12.

> "If the microcode resides in ROM, then the processor 30 can boot-up simultaneously with the hard disk 42 spin-up. Otherwise, the processor

> 30 boot-up will occur immediately after spin-up and the microcode is downloaded into processor 30."

Horning, 7:50-55.

> "In step 68, the processor 30 requests the disk data transfer unit 38 to retrieve the SSD memory 58, or more generally the buffer memory 50, configuration parameter values from a manufacturer specified location on the hard disk 42."

Horning, 7:59-63. *See also* Horning, 7:59-8:2.

> "Referring to FIG. 5B, in step 324 the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. In step 328, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer data from the SSD memory 58 to the host 19. The requested data is transferred from the SSD memory 58 through the cache/SSD data transfer unit 46 a sector at a time."

Horning, 13:12-22.

> "In this step the processor 30 determines whether or not a current version of the data to be read is in the cache 54. If so, then in step 344, the processor assigns the address of the data cache location to the variable, -"cache_loc." In step 348, the processor 30 configures the cache/SSD data transfer unit 46 to read "blk_cnt" data blocks from the cache 54 and transfer them to the buffer controller 34 without any modification or formatting. In step 352, the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_cnt" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. Referring to FIG. SC, in step 356, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer the data blocks from the cache 54 to the host 18."

Horning, 13:47-62, Fig. 5A-5C.

> "In step 364, the processor 30 determines a cache location where the data can be written from the hard disk 42 and assigns the address of this location to "cache__loc." In step 368, the processor 30 configures the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer "blk_cnt" number of data blocks from the disk data transfer unit 38 to the cache 54. Note, in this step, the cache/SSD data transfer unit 46 is

> configured to write the data to the cache 54 without any modification or formatting. In step 372, the processor 30 waits for an interrupt from the disk data transfer unit 38 indicating the seek command has completed and successfully located the data to be read. In step 376, the processor instructs the disk transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 42 to transfer the data located on the hard disk 42 to the cache 54. Preferably after a predetermined number of data bytes have been transferred to the cache 54, the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 will commence multiplexing the data transfer to the cache 54 with an additional data transfer of this same data from the cache 54 to the host 18."

Horning, 14:1-23, Fig. 5A-5C.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Horning, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

> "The solid state buffer memory, normally used as a data cache in a conventional hard disk drive, in accordance with the present invention, is partitioned into two portions one portion remains as the hard disk data cache as in the conventional hard disk drive configuration while the other portion is used as a memory for an SSD such that the SSD memory is considered by the host to be an entirely separate storage device from that of the hard disk."

Horning, 2:46-55.

> "The hard disk 42 data source provides persistent or nonvolatile data storage together with the necessary data accessing mechanisms and logic for reading and writing data. The buffer memory 50 is preferably solid state memory providing data storage plus data transfer rates much higher than the hard disk 42, in fact, preferably rates greater than the transfer rate of the host 18. Thus, both the cache memory 54 and the SSD memory 58 are intended to supply storage for those data items that are accessed frequently by the host 18. The cache memory 54 stores frequently accessed copies of hard disk 42 data items, preferably without any control

> bits relating to hard disk 42 data formatting. The SSD memory 58 stores frequently accessed data items in a "disk-like" format."

Horning, 5:17-31.

> "If the data transfer is directed to a hard disk 42 location, then preferably the transfer must be through the cache memory 54 regardless of whether the host request is a read or write. That is, if a data read is requested then a- valid copy of the requested data must either reside in the cache memory 54 or be transferred into the cache memory 54 from the hard disk 42 via the disk data transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 46. In any case, the requested data is subsequently transferred from the cache memory 54 to the host 18, via the cache/SSD data transfer unit 46, the buffer controller 34 and the host interface 26."

Horning, 6:33-45.

> "Of course, there are well known caching strategies that can be employed to facilitate these data transfers."

Horning, 6:52-54. *See also* Horning, 6:54-7:12.

> "If the microcode resides in ROM, then the processor 30 can boot-up simultaneously with the hard disk 42 spin-up. Otherwise, the processor 30 boot-up will occur immediately after spin-up and the microcode is downloaded into processor 30."

Horning, 7:50-55.

> "In step 68, the processor 30 requests the disk data transfer unit 38 to retrieve the SSD memory 58, or more generally the buffer memory 50, configuration parameter values from a manufacturer specified location on the hard disk 42."

Horning, 7:59-63. *See also* Horning, 7:59-8:2.

> "Referring to FIG. 5B, in step 324 the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. In step 328, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer data from the SSD memory 58 to the host 19. The requested data is transferred from the SSD memory 58 through the cache/SSD data transfer unit 46 a sector at a time."

Horning, 13:12-22.

> "In this step the processor 30 determines whether or not a current version of the data to be read is in the cache 54. If so, then in step 344, the processor assigns the address of the data cache location to the variable, -"cache_loc." In step 348, the processor 30 configures the cache/SSD data transfer unit 46 to read "blk_cnt" data blocks from the cache 54 and transfer them to the buffer controller 34 without any modification or formatting. In step 352, the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_cnt" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. Referring to FIG. SC, in step 356, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer the data blocks from the cache 54 to the host 18."

Horning, 13:47-62, Fig. 5A-5C.

> "In step 364, the processor 30 determines a cache location where the data can be written from the hard disk 42 and assigns the address of this location to "cache__loc." In step 368, the processor 30 configures the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer "blk_cnt" number of data blocks from the disk data transfer unit 38 to the cache 54. Note, in this step, the cache/SSD data transfer unit 46 is configured to write the data to the cache 54 without any modification or formatting. In step 372, the processor 30 waits for an interrupt from the disk data transfer unit 38 indicating the seek command has completed and successfully located the data to be read. In step 376, the processor instructs the disk transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 42 to transfer the data located on the hard disk 42 to the cache 54. Preferably after a predetermined number of data bytes have been transferred to the cache 54, the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 will commence multiplexing the data transfer to the cache 54 with an additional data transfer of this same data from the cache 54 to the host 18."

Horning, 14:1-23, Fig. 5A-5C.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Horning, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "In particular, since most of the information for anticipating future data requirements of host applications reside in the host, for most efficient cache use, the host must make decisions as to what data should be retained in the cache."

Horning, 1:31-35.

> "The embodiment described hereinabove is further intended to explain the best mode presently known of practicing the invention and to enable other skilled in the art to utilized the invention is such, or other embodiments, and with the various modifications required by their particular application or uses of the invention. It is intended that the appended claims be construed to include alternative embodiments to the extent permitted by the prior art."

Horning, 14:47-55.

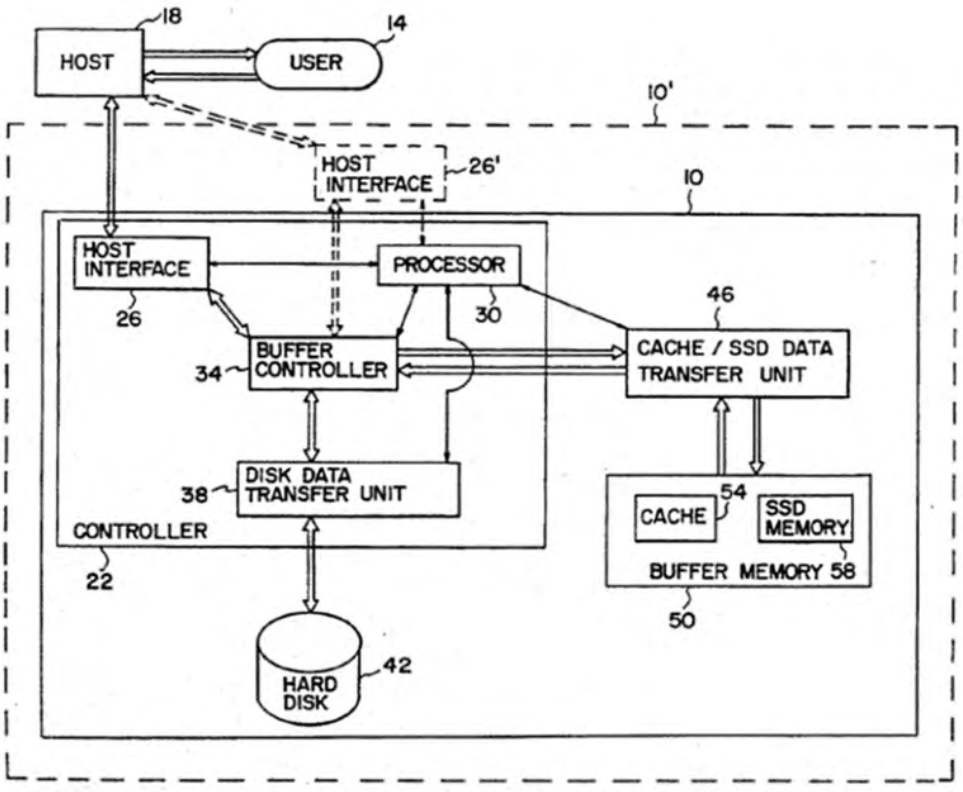
3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Horning, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See Claims 1.3, and 1.4 above.

See also



Horning, Fig. 1. *See also* Fig. 4B, 4C, 5B, 5C.

"The disk controller, therefore, is left with the responsibility of managing the data transfer between the cache, the hard disk and the

> host. In such cases the controller usually supplies a relatively simple, and less than optimal, data caching strategy such as the strategy where the most recently accessed data is always stored in the cache regardless of the access frequency."

Horning, 1:42-49. *See also* Horning, 2:14-26, 3:50-51, 4:4-6, 5:5-17, 5:42-6:10, 6:37-7:18, 10:37-14:29.

4. The method of claim 1, further comprising updating the list of boot data.	Horning, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.1 above.	

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Horning, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Horning, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.1 and 4 above.	

Appendix A7
Invalidity of U.S. Patent 7,181,608 based on Horning

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Horning, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1 (Preamble) above.	

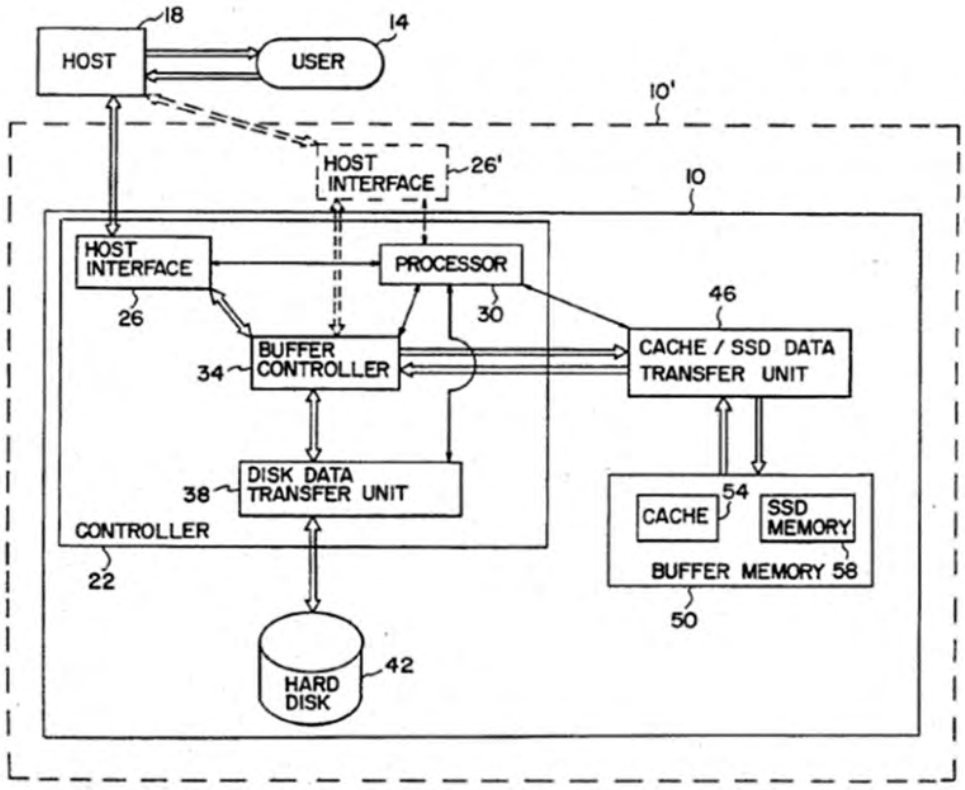
7.1 a digital signal processor (DSP) or controller;	Horning, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also



Horning, Fig. 1. *See also* Fig. 4B, 4C, 5B, 5C

> "The disk controller, therefore, is left with the responsibility of managing the data transfer between the cache, the hard disk and the host. In such cases the controller usually supplies a relatively simple, and less than optimal, data caching strategy such as the strategy where the most

recently accessed data is always stored in the cache regardless of the access frequency."

Horning, 1:42-49. *See also* Horning, 2:14-26, 3:50-51, 4:4-6, 5:5-17, 5:42-6:10, 6:37-7:18, 10:37-14:29.

7.2 a cache memory device; and;	Horning, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Horning, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Horning, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Horning, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Horning, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> "In particular, since most of the information for anticipating future data requirements of host applications reside in the host, for most efficient cache use, the host must make decisions as to what data should be retained in the cache."

Horning, 1:31-35.

> "The embodiment described hereinabove is further intended to explain the best mode presently known of practicing the invention and to enable other skilled in the art to utilized the invention is such, or other embodiments, and with the various modifications required by their particular application or uses of the invention. It is intended that the appended claims be construed to include alternative embodiments to the

extent permitted by the prior art." Horning, 14:47-55

9.1 maintaining a list of application data associated with an application program;	Horning, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Horning, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Horning, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> "In particular, since most of the information for anticipating future data requirements of host applications reside in the host, for most efficient cache use, the host must make decisions as to what data should be retained in the cache."

Horning, 1:31-35.

> "The embodiment described hereinabove is further intended to explain the best mode presently known of practicing the invention and to enable other skilled in the art to utilized the invention is such, or other embodiments, and with the various modifications required by their particular application or uses of the invention. It is intended that the appended claims be construed to include alternative embodiments to the extent permitted by the prior art."

Horning, 14:47-55.

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Horning, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

> "The hard disk 42 data source provides persistent or nonvolatile data storage together with the necessary data accessing mechanisms and logic for reading and writing data. The buffer memory 50 is preferably solid state memory providing data storage plus data transfer rates much higher than the hard disk 42, in fact, preferably rates greater than the transfer rate of the host 18. Thus, both the cache memory 54 and the SSD memory 58 are intended to supply storage for those data items that are accessed frequently by the host 18. The cache memory 54 stores frequently accessed copies of hard disk 42 data items, preferably without any control bits relating to hard disk 42 data formatting. The SSD memory 58 stores frequently accessed data items in a "disk-like" format."

Horning, 5:17-31.

> "Depending on the dual disk drive 10 configuration, the microcode instructions for the processor 30 can reside in either a read only memory (ROM) associated with processor 30 or, alternatively, the microcode can reside on the hard disk 42."

Horning, 7:46-50, Fig. 2.

> "In step 68, the processor 30 requests the disk data transfer unit 38 to retrieve the SSD memory 58, or more generally the buffer memory 50, configuration parameter values from a manufacturer specified location on the hard disk 42."

Horning, 7:59-63. *See also* Horning, 7:59-8:2.

> "The configuring is similar to step 220 in Fig. 4 in that it includes providing the cache/SSD data transfer unit 46 with: (i) the number of blocks to be read, "blk_cnt," (ii) the initial header location address, in the SSD memory 58 where the data is to be read as determined from the physical address established in step 312 above, (iii) the expected value of the sector header at this location, and (iv) a signal indicating that the data to be received from the SSD memory 58 is to be transferred to the host 18 without header and error correction bits."

Horning, 13:2-12.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Horning, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

> "Referring to FIG. 5B, in step 324 the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. In step 328, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer data from the SSD memory 58 to the host 19. The requested data is transferred from the SSD memory 58 through the cache/SSD data transfer unit 46 a sector at a time."

Horning, 13:12-22.

> "In this step the processor 30 determines whether or not a current version of the data to be read is in the cache 54. If so, then in step 344, the processor assigns the address of the data cache location to the variable, -"cache_loc." In step 348, the processor 30 configures the cache/SSD data transfer unit 46 to read "blk_cnt" data blocks from the cache 54 and transfer them to the buffer controller 34 without any modification or formatting. In step 352, the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_cnt" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. Referring to FIG. SC, in step 356, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer the data blocks from the cache 54 to the host 18."

Horning, 13:47-62, Fig. 5A-5C.

> "In step 364, the processor 30 determines a cache location where the data can be written from the hard disk 42 and assigns the address of this location to "cache__loc." In step 368, the processor 30 configures the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer "blk_cnt" number of data blocks from the disk data transfer unit 38 to the

cache 54. Note, in this step, the cache/SSD data transfer unit 46 is configured to write the data to the cache 54 without any modification or formatting. In step 372, the processor 30 waits for an interrupt from the disk data transfer unit 38 indicating the seek command has completed and successfully located the data to be read. In step 376, the processor instructs the disk transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 42 to transfer the data located on the hard disk 42 to the cache 54. Preferably after a predetermined number of data bytes have been transferred to the cache 54, the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 will commence multiplexing the data transfer to the cache 54 with an additional data transfer of this same data from the cache 54 to the host 18."

Horning, 14:1-23, Fig. 5A-5C.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Horning, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Horning, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Horning, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Horning, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.3, 1.4, and 15 above.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Horning, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Horning, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Horning, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Horning, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Horning, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Horning, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.4 above.	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Horning, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 4, 10 and 11 above.	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Horning, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 15 and 22 above.	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Horning, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 16 and 22 above.	

27.1 a boot device..	Horning, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1 above.	

27.2 a processor..	Horning, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.2 above.	

27.3 cache memory; and.	Horning, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.3 and 1.4 above.	

27.4 non-volatile memory for storing logic code for use by the processor,..	Horning, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.1 and 1.3 above.	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Horning, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.1 above.	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Horning, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.3 above.	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Horning, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.4 above.	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Horning, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 4, 10, and 11 above.	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Horning, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 15 and 27 above.	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Horning, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 16 and 27 above.	

Appendix A8
Invalidity of U.S. Patent 7,181,608 based on Hovis

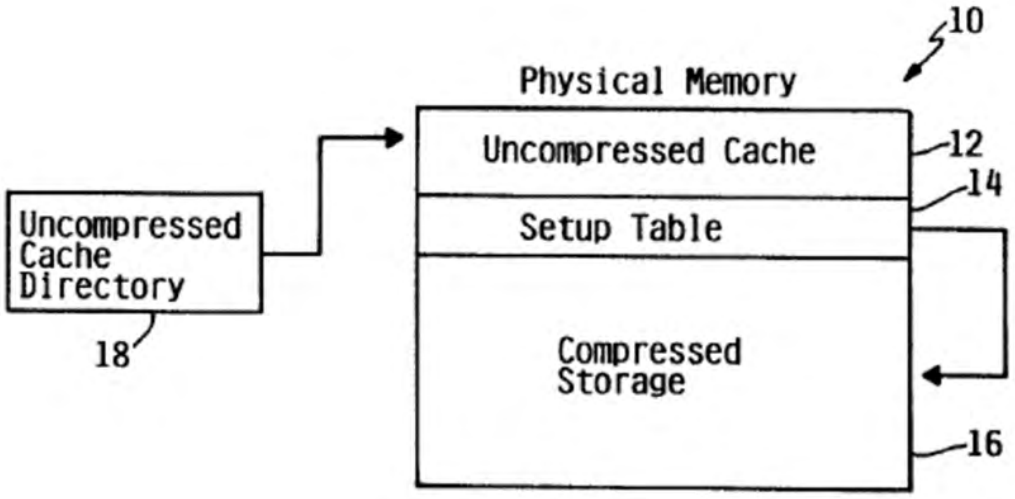
U.S. Patent No. 5,812,817 to Hovis ("Hovis") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Hovis, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:



Hovis, Fig. 1. Hovis, 1:39-51, 2:18-24, 2:34-43.

> "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15.

1.1 maintaining a list of boot data used for booting a computer system;	Hovis, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

> "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15. *See also* Fig. 2, 3:23-38.

> "When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for indicating where the compressed data resides in the compressed storage 16. This data is sent to the compression engine which in turn uses the addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 3:63-4:4.

> "Also, the directory for the uncompressed cache must be updated when new locations are added or when old (the least recently used) locations are removed to compressed memory 16."

Hovis, 4:18-21.

> "If the cache was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the cache."

Hovis, 4:40-44.

1.2 initializing a central processing unit of the computer system;	Hovis, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Hovis, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

> "The architecture includes a cache section, a setup table, and a compressed storage, all of which are partitioned from a computer memory. The cache section is used for storing uncompressed data and is a fast access memory for data which is frequently referenced. The compressed storage is used for storing compressed data. The setup table is used for specifying locations of compressed data stored within the compressed storage. A high speed uncompressed cache directory is coupled to the memory for determining if data is stored in the cache section or compressed storage and for locating data in the cache."

Hovis, Abstract. Hovis, 1:39-51, 2:18-24, 2:34-43.

> "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15.

> "When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for indicating where the compressed data resides in the compressed storage 16. This data is sent to the compression engine which in turn uses the addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 3:63-4:4.

> "If the cache was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the cache."

Hovis, 4:40-44.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Hovis, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

> "The architecture includes a cache section, a setup table, and a compressed storage, all of which are partitioned from a computer memory. The cache section is used for storing uncompressed data and is a fast access memory for data which is frequently referenced. The compressed storage is used for storing compressed data. The setup table is used for specifying locations of compressed data stored within the compressed storage. A high speed uncompressed cache directory is coupled to the memory for determining if data is stored in the cache section or compressed storage and for locating data in the cache."

Hovis, Abstract. Hovis, 1:39-51, 2:18-24, 2:34-43.

> "This invention provides a hardware assisted compression architecture that significantly reduces the typical latency to processor memory as compared to conventional compression techniques. Instead of compressing the entire contents of memory, which adds to the memory latency of all accesses, the present invention reserves a portion of memory as an uncompressed cache storage. Since most memory references are to a relatively small percentage of the stored data, which is

> preferably stored within the cache, the present invention typically avoids accesses that require additional delay to compress or decompress data while increasing memory capacity."

Hovis, 2:5-16.

> "In order to improve performance, the uncompressed cache directory typically has a fast access time. Within the general scheme of processor memory accesses, most memory accesses fall within a small range of the total available memory storage. A memory architecture, according to the present invention, can be used with a most recently used control scheme to maintain the most active segments of memory within the uncompressed storage cache 12. The directory 18 for uncompressed storage preferably must have a short access time in comparison to the overall access time of the uncompressed cache 12 in order to minimally impact typical accesses to memory."

Hovis, 2:44-55.

> "Since most accesses will be to uncompressed cache, additional performance degradations associated with going through compression are usually not encountered. This is true even with the proportionately small uncompressed cache."

Hovis, 2:60-63.

> "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15.

> "Access latency over conventional systems without compression is typically only degraded by the amount of time it takes to access the

uncompressed cache directory 18."

Hovis, 3:58-60.

> "When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for indicating where the compressed data resides in the compressed storage 16. This data is sent to the compression engine which in turn uses the addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 3:63-4:4.

> "If the cache was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the cache."

Hovis, 4:40-44.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Hovis, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Hovis, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.3, and 1.4 above.	

4. The method of claim 1, further comprising updating the list of boot data.	Hovis, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See Claim 1.1 above.

See also

> "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15. *See also* Fig. 2, 3:23-38.

> "Also, the directory for the uncompressed cache must be updated when new locations are added or when old (the least recently used) locations are removed to compressed memory 16."

Hovis, 4:18-21.

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Hovis, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Hovis, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Hovis, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Hovis, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.2, 1.3, and 1.4 above.	

7.2 a cache memory device; and;	Hovis, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Hovis, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Hovis, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Hovis, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Hovis, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.	

9.1 maintaining a list of application data associated with an application program;	Hovis, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

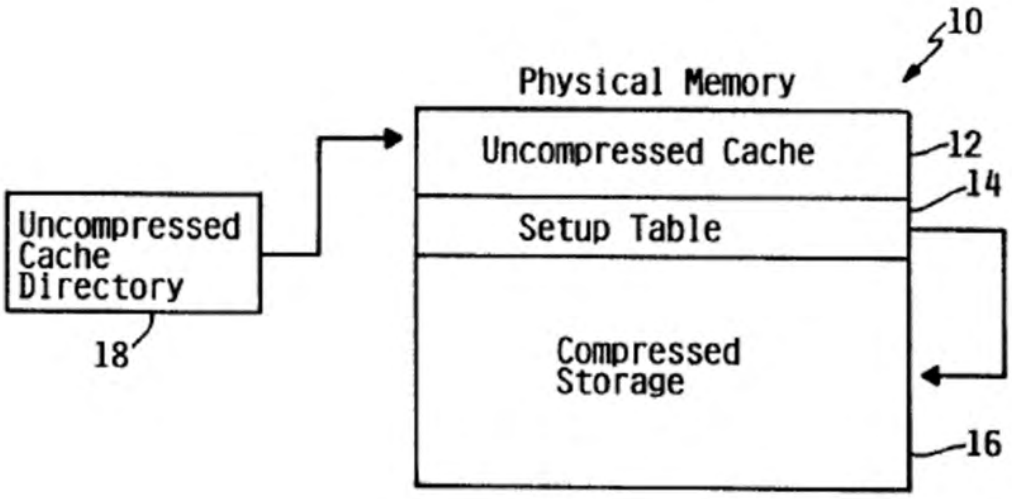
9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Hovis, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Hovis, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.4, 2, and 8 above.	

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Hovis, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:



Hovis, Fig. 1. Hovis, 2:34-43.

> "The architecture includes a cache section, a setup table, and a compressed storage, all of which are partitioned from a computer memory. The cache section is used for storing uncompressed data and is a fast access memory for data which is frequently referenced. The compressed storage is used for storing compressed data. The setup table is used for specifying locations of compressed data stored within the compressed storage."

Hovis, Abstract. Hovis, 1:39-51, 2:18-24.

> "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15. *See also* Fig. 2, 3:23-38.

> "Fig. 4 shows a basic dataflow structure of the memory 10 with a hardware assist compression engine 28. A memory control 24 interfaces with the memory 10, compression engine 28, and processor units 22. The purpose of a hardware based compression engine 28 is to provide the necessary bandwidth and latency required by memory entities that reside close to the processor data/instruction units (i.e. L1, L2, L3). Typical software based compression techniques have limited bandwidth and significantly greater latency which would substantially degrade the processor performance."

Hovis, 3:39-50, Fig. 4.

> "When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for indicating where the compressed data resides in the compressed storage 16. This data is sent to the compression engine which in turn uses the addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 3:63-4:4.

> "Also, the directory for the uncompressed cache must be updated when new locations are added or when old (the least recently used) locations are removed to compressed memory 16."

Hovis, 4:18-21.

Hovis Claim 10

"The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device"

> "If the cache was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the cache."

Hovis, 4:40-44.

> "Aspects of the control of the hardware assisted memory compression can be implemented either by hardware, software (operating system, namely memory management), or a combination of both. The present invention is not dependent upon a particular hardware or software control scheme. The compression engine is preferably implemented in hardware in order to perform the compression and decompression in a timely fashion and with sufficient bandwidth."

Hovis, 5:4-11.

> "The implementation involves a hardware compression engine and the control can be via hardware, software, or a combination of both."

Hovis, 5:27-30.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Hovis, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

> "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15.

> "When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for indicating where the compressed data resides in the compressed storage 16. This data is sent to the compression engine which in turn uses the addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 3:63-4:4.

> "If the cache was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the cache."

Hovis, 4:40-44.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Hovis, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Hovis, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See Claims 1.3 and 1.4 above.

> "Within the general scheme of processor memory accesses, most memory accesses fall Within a small range of the total available memory storage. A memory architecture, according to the present invention, can be used with a most recently used control scheme to maintain the most active segments of memory within the uncompressed storage cache 12. The directory 18 for uncompressed storage preferably must have a short access time in comparison to the overall access time of the uncompressed cache 12 in order to minimally impact typical accesses to memory."

Hovis, 2:45-55.

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Hovis, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See Claims 1.3 and 1.4 above.

> "A compression technique is typically used With this memory architecture. For example, loss-less compression techniques used with the present invention could provide a two times or greater improvement in real memory capacity. This gain is dependent on both data patterns and the chosen compression algorithm. The present invention is not dependent on any particular compression algorithm; it can use any lossless compression algorithm in general."

Hovis, 2:23-30.

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Hovis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

> "A compression technique is typically used with this memory architecture. For example, loss-less compression techniques used with the present invention could provide a two times or greater improvement in real memory capacity. This gain is dependent on both data patterns and the chosen compression algorithm. The present invention is not dependent on any particular compression algorithm; it can use any lossless compression algorithm in general."

Hovis, 2:23-30.

> "The address results in an access to the compressed storage 16 Which in turn results in compressed data being accessed and processed by the compression engine 28, Which performs compression and decompression on the data."

Hovis, 3:8-12.

> "FIG. 4 shows a basic data flow structure of the memory 10 with a hardware assist compression engine 28. A memory control 24 interfaces With the memory 10, compression engine 28, and processor units 22. The purpose of a hardware based compression engine 28 is to provide the necessary bandwidth and latency required by memory entities that reside close to the processor data/instruction units (i.e. L1,L2,L3)."

Hovis, 3:41-46

> "The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 4:2-4.

> "If the cache is full, then the system must do a cast out and transfer the least recently used data element in the cache to the compressed storage (48), Which requires sending the data to the compression engine for compressing (46). The system then updates the data element in the cache (50) and continues as if the cache Was not full. If the cache Was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38)."

Hovis, 4:34-44.

> "Aspects of the control of the hardware assisted memory compression can be implemented either by hardware, software (operating system, namely memory management), or a combination of both. The present invention is not dependent upon a particular hardware or software control scheme. The compression engine is preferably implemented in hardware in order to perform the compression and decompression in a timely fashion and With sufficient bandwidth."

Hovis, 5:4-11.

> "The implementation involves a hardware compression engine and the control can be via hardware, software, or a combination of both."

Hovis, 5:27-30.

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Hovis, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Hovis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Hovis, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Hovis, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Hovis, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Hovis, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Hovis, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Hovis, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Hovis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Hovis, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1 above.	

27.2 a processor..	Hovis, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.2 above.	

27.3 cache memory; and.	Hovis, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.3 and 1.4 above.	

27.4 non-volatile memory for storing logic code for use by the processor,..	Hovis, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.1 and 1.3 above.	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Hovis, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.1 above.	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Hovis, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.3 above.	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Hovis, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.4 above.	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Hovis, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 4, 10, and 11 above.	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Hovis, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 15 and 27 above.	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Hovis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 16 and 27 above.	

Appendix A9
Invalidity of U.S. Patent 7,181,608 based on Ingvar

G.B. Patent No. 2276257 to Ingvar ("Ingvar") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A9
Invalidity of U.S. Patent 7,181,608 based on Ingvar

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Ingvar, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

> "A method for enabling the transfer of stored firmware during start-up of a computer, comprising the steps of reading configuration information stored in a air configuration table in the first memory device (40), transferring selected software modules from the first memory device to a second memory device (130, 60, 50), and storing the selected software modules at selected address areas in the second memory device; in all accordance with the configuration information."

Ingvar, Abstract.

> "In order to provide quicker access to the program modules, these modules are normally copied from the PROM-packages to a system-contained main memory which has shorter access times."

Ingvar at 6.

1.1 maintaining a list of boot data used for booting a computer system;	Ingvar, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

> "A method for enabling the transfer of stored firmware during start-up of a computer, comprising the steps of reading configuration information stored in a air configuration table in the first memory device (40), transferring selected software modules from the first memory device to a second memory device (130, 60, 50), and storing the selected software modules at selected address areas in the second memory device; in all accordance with the configuration information."

Ingvar, Abstract.

> "The resultant configuration data or information can be stored in a configuration table which will thus disclose those program modules which are required by the computer system in respect of the current configuration of said system."

Ingvar at 11-12. *See also* Ingvar at 12, ¶¶ 1-2.

> "The area c may include the aforesaid configuration table containing information as to which software modules shall be used, and information as to which addresses and software modules respectively shall be work-stored."

Ingvar at 13.

> "In Step S150, the startup program can use the information in the configuration table to decide which of the program modules shall be used in the computer system and to obtain information as to where, i.e. at which addresses, respective software modules shall be placed in the working memory 130 for future use, until the computer unit is stopped and restarted."

Ingvar at 14-15.

> "Step S250 According to one embodiment of the invention, there is inserted in Step S250 configuration information which may also be stored in the first memory device. The configuration information may have the form of a data table which discloses, among other things, the memory addresses at which the various software modules shall be stored when decompressed. The table may also include information as to which software modules shall be work-stored in the working memory and/or the addresses at which the modules shall be work-stored when carrying out the first method according to the invention, as described above."

Ingvar at 18.

1.2 initializing a central processing unit of the computer system;	Ingvar, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

> "According to one aspect on the present invention there is provided a method for use when starting up a computer system arrangement, the arrangement comprising at least one computer unit and at least one first memory device which includes at least one first data block with information stored as firmware, the method including the step of initiating the start of the computer unit and the step of transferring software modules from the first memory device to a second memory device, characterized by the steps of reading changeable configuration information stored in a configuration table in the first memory device; transferring selected modules among said software modules from the first memory device to the second memory device, and storing the selected software modules at selected address areas in the second memory device in accordance with the configuration information."

Ingvar at 10.

> "When starting-up the computer unit 20, the data processing device 30 reads information contained in a memory device 40."

Ingvar at 10.

> "Step S120 and S130 In Step S120, the data processing unit 30 included in the computer unit executes a first startup program. This startup program can be executed and when executed, the first startup program may include a routine which requires a check to be made as to which peripheral units are included in the hardware included in the computer system, as illustrated by Step S130 in Fig. 2."

Ingvar at 11.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Ingvar, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

> "A method for enabling the transfer of stored firmware during start-up of a computer, comprising the steps of reading configuration information stored in a air configuration table in the first memory device (40), transferring selected software modules from the first memory device to a second memory device (130, 60, 50), and storing the selected software modules at selected address areas in the second memory device; in all accordance with the configuration information."

Ingvar, Abstract.

> "The memory device 40 may include firmware, such as startup software and BIOS program modules. According to the invention, certain software modules may be stored in a compressed form in the memory device 40. A compressed data packet which includes one or more software modules will take up less memory space than that taken up by a corresponding amount of data or information stored in an uncompressed state, as is well known to the person skilled in this art."

Ingvar at 10.

> "The data processing device 30 can avail itself of the memory device 40 precisely when starting-up the computer unit 20, when working with addresses within the range between mid~addr and max~addr, and can avail itself of a memory device 50 when working with data at addresses which lie within the range between min~addr and mid~addr. The memory

device 50 may be a volatile memory."

Ingvar at 10.

"Step S140 also involves copying the software modules a, c and e to a memory area 125 in the memory device 50 (Fig. 3B). This means that the executing program can copy itself and then continue to execute from the memory area 125."

Ingvar at 14.

"Steps S220 and S230 The selected software is compressed in Step S220 and is stored in Step S230 in a selected memory area in the first memory device 40, which will retain its data content in the absence of an applied voltage."

Ingvar at 18.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Ingvar, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

> "Step 5140 In Step S140, the first startup program summons a decompression or decompaction program which unpacks or decompresses the firmware modules that are given in the configuration table and that are stored in a compressed state. According to another embodiment of the inventive method, the first decompression program decompresses all firmware modules stored in the memory device 40, wherein the choice of those modules that shall be used is made in accordance with the configuration table after said decompression."

Ingvar at 12.

> "Step S240 In Step S240, there is inserted a decompression program, which can also be stored in the first memory device 40."

Ingvar at 18.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Ingvar, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "Another object of the present invention is to-enable the selection of which software modules shall be placed in which memory positions with the aid of software."

Ingvar at 7.

> "According to one aspect on the present invention there is provided a method for use when starting up a computer system arrangement, the arrangement comprising at least one computer unit and at least one first memory device which includes at least one first data block with information stored as firmware, the method including the step of initiating the start of the computer unit and the step of transferring software modules from the first memory device to a second memory device, characterized by the steps of reading changeable configuration information stored in a configuration table in the first memory device; transferring selected modules among said software modules from the first memory device to the second memory device, and storing the selected software modules at selected address areas in the second memory device in accordance with the configuration information."

Ingvar at 10.

> "The invention also relates to a method of storing firmware in a computer unit so that software modules stored as firmware can be easily made movable to selected addresses in the working memory 130."

Ingvar at 18.

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Ingvar, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.3, and 1.4 above.	

4. The method of claim 1, further comprising updating the list of boot data.	Ingvar, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

See Claim 1.1 above.

See also

> "A further object of the present invention is to provide a method to enable program modules or data tables which are stored as firmware when starting-up the computer system to be repositioned such that the remaining unoccupied memory space can be readily utilized."

Ingvar at 7.

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Ingvar, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Ingvar, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.1 and 4 above.	

Appendix A9
Invalidity of U.S. Patent 7,181,608 based on Ingvar

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Ingvar, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Ingvar, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.2, 1.3, and 1.4 above.	

7.2 a cache memory device; and;	Ingvar, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Ingvar, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Ingvar, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Ingvar, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Ingvar, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> "Another object of the present invention is to-enable the selection of which software modules shall be placed in which memory positions with the aid of software."

Ingvar at 7.

> "According to one aspect on the present invention there is provided a method for use when starting up a computer system arrangement, the arrangement comprising at least one computer unit and at least one first memory device which includes at least one first data block with information stored as firmware, the method including the step of initiating the start of the computer unit and the step of transferring software modules from the first memory device to a second memory device, characterized by the steps of reading changeable configuration

information stored in a configuration table in the first memory device; transferring selected modules among said software modules from the first memory device to the second memory device, and storing the selected software modules at selected address areas in the second memory device in accordance with the configuration information."

Ingvar at 10.

"The invention also relates to a method of storing firmware in a computer unit so that software modules stored as firmware can be easily made movable to selected addresses in the working memory 130."

Ingvar at 18.

Appendix A9
Invalidity of U.S. Patent 7,181,608 based on Ingvar

9.1 maintaining a list of application data associated with an application program;	Ingvar, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Ingvar, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Ingvar, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> "Another object of the present invention is to-enable the selection of which software modules shall be placed in which memory positions with the aid of software."

Ingvar at 7.

> "According to one aspect on the present invention there is provided a method for use when starting up a computer system arrangement, the arrangement comprising at least one computer unit and at least one first memory device which includes at least one first data block with information stored as firmware, the method including the step of initiating the start of the computer unit and the step of transferring software modules from the first memory device to a second memory device, characterized by the steps of reading changeable configuration information stored in a configuration table in the first memory device; transferring selected modules among said software modules from the

first memory device to the second memory device, and storing the selected software modules at selected address areas in the second memory device in accordance with the configuration information."

Ingvar at 10.

"The invention also relates to a method of storing firmware in a computer unit so that software modules stored as firmware can be easily made movable to selected addresses in the working memory 130."

Ingvar at 18.

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Ingvar, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

> "A method for enabling the transfer of stored firmware during start-up of a computer, comprising the steps of reading configuration information stored in a air configuration table in the first memory device (40), transferring selected software modules from the first memory device to a second memory device (130, 60, 50), and storing the selected software modules at selected address areas in the second memory device; in all accordance with the configuration information."

Ingvar, Abstract.

> "The memory device 40 may include firmware, such as startup software and BIOS program modules. According to the invention, certain software modules may be stored in a compressed form in the memory device 40. A compressed data packet which includes one or more software modules will take up less memory space than that taken up by a corresponding amount of data or information stored in an uncompressed state, as is well known to the person skilled in this art."

Ingvar at 10.

> "The resultant configuration data or information can be stored in a configuration table which will thus disclose those program modules which are required by the computer system in respect of the current configuration of said system."

Ingvar at 11-12. *See also* Ingvar at 12, ¶¶ 1-2.

> "The area c may include the aforesaid configuration table containing information as to which software modules shall be used, and information as to which addresses and software modules respectively shall be work-stored."

Ingvar at 13.

> "In Step S150, the startup program can use the information in the configuration table to decide which of the program modules shall be used in the computer system and to obtain information as to where, i.e. at which addresses, respective software modules shall be placed in the working memory 130 for future use, until the computer unit is stopped and restarted."

Ingvar at 14-15.

> "According to this embodiment, when the configuration table is stored in a compressed state in the memory device 40, the reconfiguration command can provide access to the decompressed configuration table in the working memory 130. According to this latter embodiment of the invention, the user is able to initiate compression and storage of the modified table in the permanent memory device 40."

Ingvar at 17.

> "Steps S220 and S230 The selected software is compressed in Step S220 and is stored in Step S230 in a selected memory area in the first memory device 40, which will retain its data content in the absence of an applied voltage."

Ingvar at 18.

> "Step S250 According to one embodiment of the invention, there is inserted in Step S250 configuration information which may also be stored in the first memory device. The configuration information may have the form of a data table which discloses, among other things, the memory addresses at which the various software modules shall be stored when decompressed. The table may also include information as to which software modules shall be work-stored in the working memory and/or the addresses at which the modules shall be work-stored when carrying out the first method according to the invention, as described above."

Ingvar at 18.

> "According to another embodiment of the invention, the software

> modules are stored in the permanent memory device 40 in an uncompressed state, together with the aforesaid configuration information and startup program."
>
> Ingvar at 19.

Ingvar Claim 10

"The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device"

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Ingvar, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

> "Step 5140 In Step S140, the first startup program summons a decompression or decompaction program which unpacks or decompresses the firmware modules that are given in the configuration table and that are stored in a compressed state. According to another embodiment of the inventive method, the first decompression program decompresses all firmware modules stored in the memory device 40, wherein the choice of those modules that shall be used is made in accordance with the configuration table after said decompression."

Ingvar at 12.

> "Step S240 In Step S240, there is inserted a decompression program, which can also be stored in the first memory device 40."

Ingvar at 18.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Ingvar, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Ingvar, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Ingvar, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

See Claims 1.3 and 1.4 above.

See also

> "The memory area c may also include the aforesaid decompression program, which can be used to decompress compressed data. The memory area d may be a memory area which contains compressed software modules."

Ingvar at 13.

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Ingvar, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

See also

> "Yet another object is to reduce the amount of hardware required by a computer system, for instance such hardware as address decoders, jumpers, permanent memory sockets, etc."

Ingvar at 8.

> "The memory area c may also include the aforesaid decompression program, which can be used to decompress compressed data. The memory area d may be a memory area which contains compressed software modules."

Ingvar at 13.

> "According to this embodiment, when the configuration table is stored in a compressed state in the memory device 40, the reconfiguration command can provide access to the decompressed configuration table in the working memory 130. According to this latter embodiment of the invention, the user is able to initiate compression and storage of the modified table in the permanent memory device 40."

Ingvar at 17.

Appendix A9
Invalidity of U.S. Patent 7,181,608 based on Ingvar

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Ingvar, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Ingvar, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above.	

22.1 maintaining a list of boot data used for booting a computer system;.	Ingvar, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.1 above.	

22.2 initializing a central processing unit of the computer system;.	Ingvar, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.2 above.	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Ingvar, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.3 above.	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Ingvar, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.4 above.	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Ingvar, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 4, 10 and 11 above.	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Ingvar, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 15 and 22 above.	

Appendix A9
Invalidity of U.S. Patent 7,181,608 based on Ingvar

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Ingvar, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 16 and 22 above.	

27.1 a boot device..	Ingvar, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1 above.	

27.2 a processor..	Ingvar, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.2 above.	

27.3 cache memory; and.	Ingvar, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.3 and 1.4 above.	

27.4 non-volatile memory for storing logic code for use by the processor,..	Ingvar, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.1 and 1.3 above.	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Ingvar, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.1 above.	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Ingvar, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.3 above.	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Ingvar, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.4 above.	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Ingvar, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 4, 10, and 11 above.	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Ingvar, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 15 and 27 above.	

Appendix A9
Invalidity of U.S. Patent 7,181,608 based on Ingvar

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Ingvar, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 16 and 27 above.	

Appendix A10
Invalidity of U.S. Patent 7,181,608 based on Kikinis

PCT Application No. WO 94/19768 to Kikinis ("Kikinis") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A10
Invalidity of U.S. Patent 7,181,608 based on Kikinis

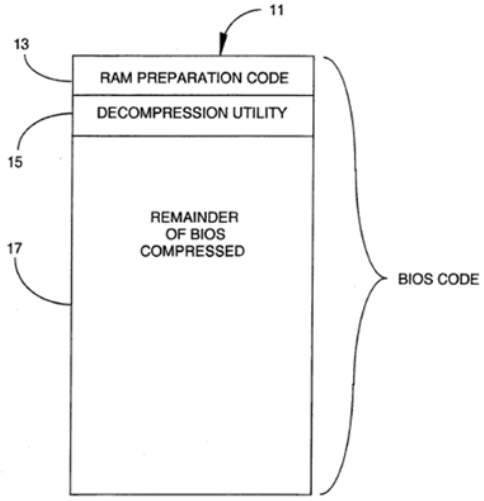
1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Kikinis, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

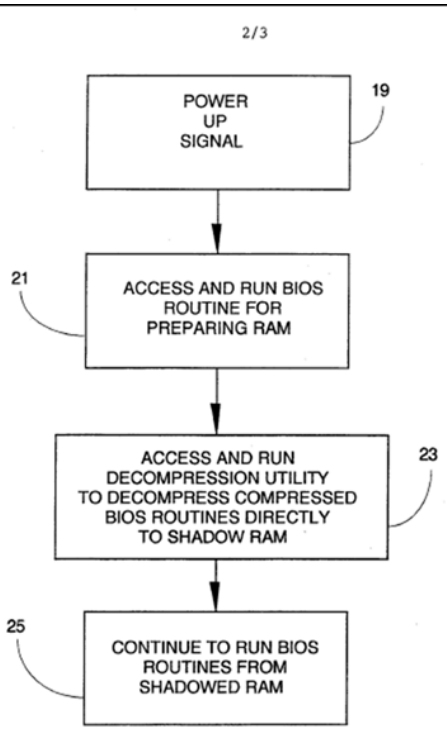
Kikinis discloses this claim limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract



Kikinis, Fig. 1



Kikinis, Fig 2

See also Kikinis Fig. 3.

1.1 maintaining a list of boot data used for booting a computer system;	Kikinis, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this claim limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

> In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the

> CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

> Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.
>
> Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.
>
> After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

See also Kikinis Fig. 1-3.

1.2 initializing a central processing unit of the computer system;	Kikinis, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this claim limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> As is well known in the art, a computer system, to be of any use, must comprise all the computer hardware, such as the CPU, memory devices, communication buses, and so forth. There must also be instruction sets (programs/software) for the CPU to follow to accomplish tasks. The programmed information (application software) is typically stored on an internal or peripheral memory device to be accessed by the CPU as needed.

Kikinis, 1

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress

it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

In most BIOS systems for general purpose computers, the BIOS is stored in an EPROM device as described in the background section above. Upon power up the BIOS initializes the system, doing basic tasks like accessing and checking the operation of on-board random-access memory RAM, and typically, somewhere during the initialization, at least a part of the BIOS code is copied (the BIOS copies itself) into a portion of the on-board RAM.

Kikinis, 4

In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

Fig. 1 is a diagrammatical representation of a compressed BIOS 11 according to the present invention. There are three different portions of the code. Portion 13 is code to perform all operations to initialize and test the system RAM, and make it ready for use, and is a familiar portion of conventional BIOS routines. This portion in some applications needs to perform such functions as initializing and testing a memory controller and cache controllers and cache memory. Portion 15 is a decompression utility. Portion 17 represents the balance of the BIOS code in compressed form. It will be apparent to those with skill in the art that there are a number of compression schemes and related decompression routines that might be used.

Kikins 5

Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during

> which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.
>
> Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.
>
> After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

See also Kikinis Fig. 1-3.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Kikinis, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this claim limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> As is well known in the art, a computer system, to be of any use, must comprise all the computer hardware, such as the CPU, memory devices, communication buses, and so forth. There must also be instruction sets (programs/software) for the CPU to follow to accomplish tasks. The programmed information (application software) is typically stored on an internal or peripheral memory device to be accessed by the CPU as needed.

Kikinis, 1

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer

> having a CPU microprocessor, comprising a programmable non-volatile memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

> In most BIOS systems for general purpose computers, the BIOS is stored in an EPROM device as described in the background section above. Upon power up the BIOS initializes the system, doing basic tasks like accessing and checking the operation of on-board random-access memory RAM, and typically, somewhere during the initialization, at least a part of the BIOS code is copied (the BIOS copies itself) into a portion of the on-board RAM.

Kikinis, 4

> In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

> The present invention comprises a means of compressing at least a significant portion of the BIOS code, storing all of the BIOS code, including the compressed portion, in EPROM, and releasing the compressed code on powerup, so all of the code is available for the computer to use. Also on powerup, the entire code is shadowed to RAM

Kikinis, 4-5

> Fig. 1 is a diagrammatical representation of a compressed BIOS 11 according to the present invention. There are three different portions of the code. Portion 13 is code to perform all operations to initialize and test the system RAM, and make it ready for use, and is a familiar portion of conventional BIOS routines. This portion in some

applications needs to perform such functions as initializing and testing a memory controller and cache controllers and cache memory. Portion 15 is a decompression utility. Portion 17 represents the balance of the BIOS code in compressed form. It will be apparent to those with skill in the art that there are a number of compression schemes and related decompression routines that might be used.

Kikins 5

Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.

Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.

After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

One means by which the BIOS code may be compressed is based on the fact that BIOS routines, as is common in most other coded instruction sets, make use of frequently repeated code sequences. EPROMs used for BIOS are typically byte-wide devices, that is, the device can store "words" of 8 bits. A sixteen bit word requires, then, two lines of BIOS code.

Kikinis, 6

See also Kikinis Fig. 1-3

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Kikinis, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this claim limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> As is well known in the art, a computer system, to be of any use, must comprise all the computer hardware, such as the CPU, memory devices, communication buses, and so forth. There must also be instruction sets (programs/software) for the CPU to follow to accomplish tasks. The programmed information (application software) is typically stored on an

> internal or peripheral memory device to be accessed by the CPU as needed.

Kikinis, 1

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

> In most BIOS systems for general purpose computers, the BIOS is stored in an EPROM device as described in the background section above. Upon power up the BIOS initializes the system, doing basic tasks like accessing and checking the operation of on-board random-access memory RAM, and typically, somewhere during the initialization, at least a part of the BIOS code is copied (the BIOS copies itself) into a portion of the on-board RAM.

Kikinis, 4

> In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

> The present invention comprises a means of compressing at least a significant portion of the BIOS code, storing all of the BIOS code, including the compressed portion, in EPROM, and releasing the compressed code on powerup, so all of the code is available for the computer to use. Also on powerup, the entire code is shadowed to RAM

Kikinis, 4-5

> Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.
>
> Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.
>
> After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

> Fig. 3 is a diagrammatical representation of the token decompression scheme described above. After the BIOS according to the embodiment of the invention has initialized and tested the RAM, the decompression utility is booted, and begins to read the compressed portion of the BIOS at Start 27. At 29 the decompression utility loads the first/next byte from the compressed portion of
> the EPROM BIOS. If this byte is hex FF (31), it is recognized as a token, and control goes to 33,. where the system reads the byte following the token flag. This byte is always a pointer to a code sequence.

Kikinis, 7

See also Kikinis Fig. 1-3.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Kikinis, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Kikinis, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

> Fig. 1 is a diagrammatical representation of a compressed BIOS 11 according to the present invention. There are three different portions of the code. Portion 13 is code to perform all operations to initialize and test the system RAM, and make it ready for use, and is a familiar portion of conventional BIOS routines. This portion in some applications needs to perform such functions as initializing and testing a memory controller and cache controllers and cache memory. Portion 15 is a decompression utility. Portion 17 represents the balance of the BIOS code in compressed form. It will be apparent to those with skill in the art that there are a number of compression schemes and related decompression routines that might be used.

Kikinis, 5

4. The method of claim 1, further comprising updating the list of boot data.	Kikinis, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Kikinis, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Kikinis, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Kikinis, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Kikinis, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

> Fig. 1 is a diagrammatical representation of a compressed BIOS 11 according to the present invention. There are three different portions of the code. Portion 13 is code to perform all operations to initialize and test the system RAM, and make it ready for use, and is a familiar portion of conventional BIOS routines. This portion in some applications needs to perform such functions as initializing and testing a memory controller and cache controllers and cache memory. Portion 15 is a decompression utility. Portion 17 represents the balance of the BIOS code in compressed form. It will be apparent to those with skill in the art that there are a number of compression schemes and related decompression routines that might be used.

Kikinis, 5

7.2 a cache memory device; and;	Kikinis, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Kikinis, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Kikinis, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Kikinis, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Kikinis, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.	

9.1 maintaining a list of application data associated with an application program;	Kikinis, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Kikinis, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Kikinis, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.4, 2, and 8 above.	

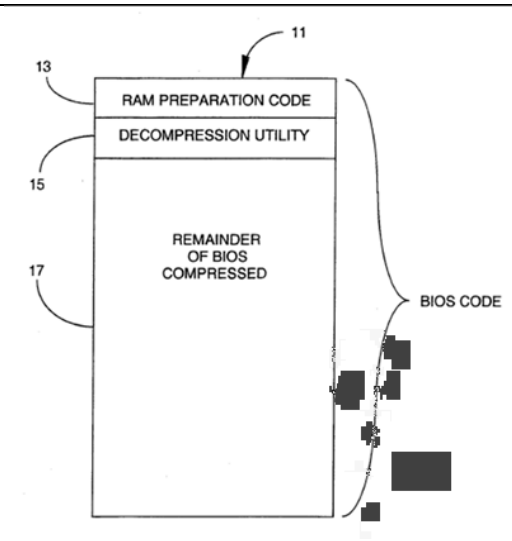
10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Kikinis, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

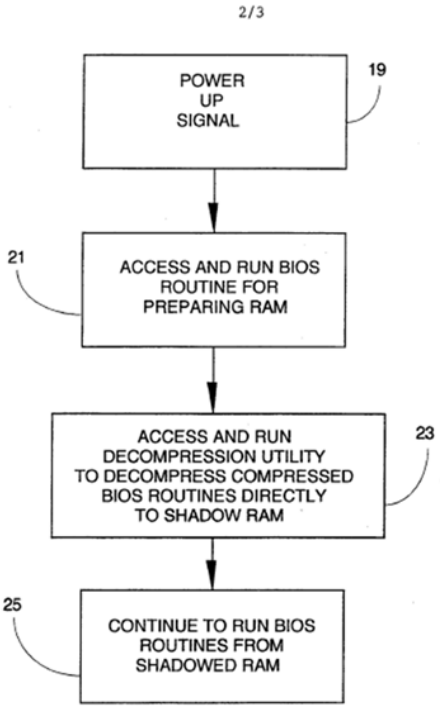
Kikinis discloses this limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract



Kikinis, Fig. 1



Kikinis, Fig 2

As is well known in the art, a computer system, to be of any use, must comprise all the computer hardware, such as the CPU, memory devices, communication buses, and so forth. There must also be instruction sets

(programs/software) for the CPU to follow to accomplish tasks. The programmed information (application software) is typically stored on an internal or peripheral memory device to be accessed by the CPU as needed.

Kikinis, 1

A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

In most BIOS systems for general purpose computers, the BIOS is stored in an EPROM device as described in the background section above. Upon power up the BIOS initializes the system, doing basic tasks like accessing and checking the operation of on-board random-access memory RAM, and typically, somewhere during the initialization, at least a part of the BIOS code is copied (the BIOS copies itself) into a portion of the on-board RAM.

Kikinis, 4

In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

The present invention comprises a means of compressing at least a significant portion of the BIOS code, storing all of the BIOS code, including the compressed portion, in EPROM, and releasing the

compressed code on powerup, so all of the code is available for the computer to use. Also on powerup, the entire code is shadowed to RAM

Kikinis, 4-5

Fig. 1 is a diagrammatical representation of a compressed BIOS 11 according to the present invention. There are three different portions of the code. Portion 13 is code to perform all operations to initialize and test the system RAM, and make it ready for use, and is a familiar portion of conventional BIOS routines. This portion in some applications needs to perform such functions as initializing and testing a memory controller and cache controllers and cache memory. Portion 15 is a decompression utility. Portion 17 represents the balance of the BIOS code in compressed form. It will be apparent to those with skill in the art that there are a number of compression schemes and related decompression routines that might be used.

Kikins 5

Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.

Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.

After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

One means by which the BIOS code may be compressed is based on the fact that BIOS routines, as is common in most other coded instruction sets, make use of frequently repeated code sequences. EPROMs used

> for BIOS are typically byte-wide devices, that is, the device can store "words" of 8 bits. A sixteen bit word requires, then, two lines of BIOS code.

Kikinis, 6

> Fig. 3 is a diagrammatical representation of the token decompression scheme described above. After the BIOS according to the embodiment of the invention has initialized and tested the RAM, the decompression utility is booted, and begins to read the compressed portion of the BIOS at Start 27. At 29 the decompression utility loads the first/next byte from the compressed portion of
> the EPROM BIOS. If this byte is hex FF (31), it is recognized as a token, and control goes to 33,. where the system reads the byte following the token flag. This byte is always a pointer to a code sequence.

Kikinis, 7

See also Kikinis Fig. 3.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Kikinis, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> As is well known in the art, a computer system, to be of any use, must comprise all the computer hardware, such as the CPU, memory devices, communication buses, and so forth. There must also be instruction sets (programs/software) for the CPU to follow to accomplish tasks. The programmed information (application software) is typically stored on an internal or peripheral memory device to be accessed by the CPU as needed.

Kikinis, 1

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a

preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

In most BIOS systems for general purpose computers, the BIOS is stored in an EPROM device as described in the background section above. Upon power up the BIOS initializes the system, doing basic tasks like accessing and checking the operation of on-board random-access memory RAM, and typically, somewhere during the initialization, at least a part of the BIOS code is copied (the BIOS copies itself) into a portion of the on-board RAM.

Kikinis, 4

In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

The present invention comprises a means of compressing at least a significant portion of the BIOS code, storing all of the BIOS code, including the compressed portion, in EPROM, and releasing the compressed code on powerup, so all of the code is available for the computer to use. Also on powerup, the entire code is shadowed to RAM

Kikinis, 4-5

Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.

Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes

> the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.
>
> After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

> Fig. 3 is a diagrammatical representation of the token decompression scheme described above. After the BIOS according to the embodiment of the invention has initialized and tested the RAM, the decompression utility is booted, and begins to read the compressed portion of the BIOS at Start 27. At 29 the decompression utility loads the first/next byte from the compressed portion of
> the EPROM BIOS. If this byte is hex FF (31), it is recognized as a token, and control goes to 33,. where the system reads the byte following the token flag. This byte is always a pointer to a code sequence.

Kikinis, 7

See also Kikinis Fig. 1-3.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Kikinis, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Kikinis, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Kikinis, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this limitation:

See Claims 1.3 and 1.4 above.

See also

> In a particular compression scheme compression is accomplished by a substituting a two line token for a longer sequence, where the longer sequence is a sequence often repeated in the BIOS. The value of the first line is a flag to the decompression routine that the next line is to be used to correlate to the longer sequence and copy that longer sequence in decompression.

Kikinis, 3

> It is also true that there are a truly large number of compression schemes that might be employed to compress a portion of the BIOS. The invention should not be limited by the specific code relationship determined to compress the BIOS code.

Kikinis, 8

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Kikinis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

See also

> In a particular compression scheme compression is accomplished by a substituting a two line token for a longer sequence, where the longer sequence is a sequence often repeated in the BIOS. The value of the first line is a flag to the decompression routine that the next line is to be used to correlate to the longer sequence and copy that longer sequence in decompression.

Kikinis, 3

> It is also true that there are a truly large number of compression schemes that might be employed to compress a portion of the BIOS. The invention should not be limited by the specific code relationship determined to compress the BIOS code.

Kikinis, 8

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Kikinis, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Kikinis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Kikinis, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Kikinis, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Kikinis, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Kikinis, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Kikinis, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Kikinis, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Kikinis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Kikinis, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Kikinis, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Kikinis, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Kikinis, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Kikinis, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Kikinis, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Kikinis, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Kikinis, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Kikinis, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 15 and 27 above	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Kikinis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 16 and 27 above	

Appendix A11
Invalidity of U.S. Patent 7,181,608 based on Krocker

U.S. Patent No. 6,073,232 to Krocker ("Krocker") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Krocker, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Krocker discloses this limitation:

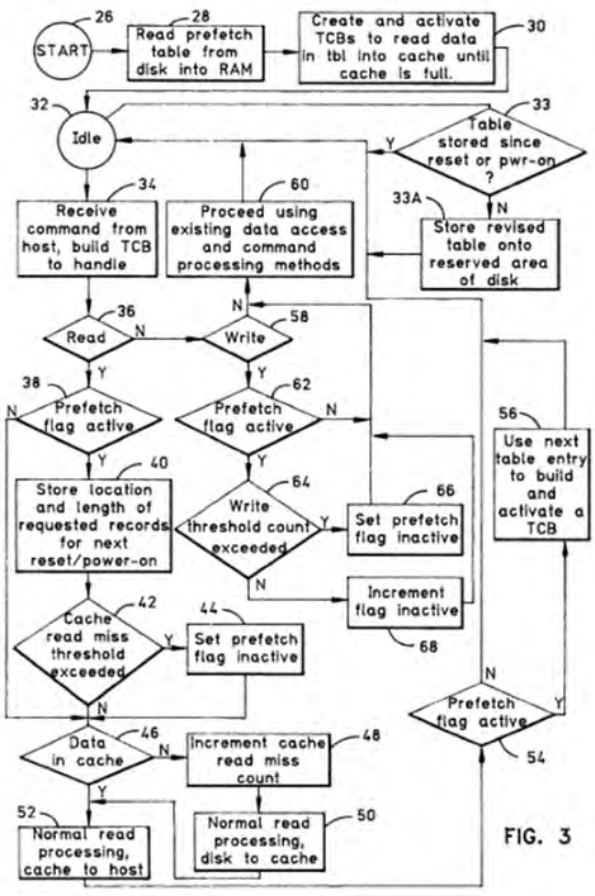


FIG. 3

Kroeker, Fig. 3.

"A method for increasing boot speed of a host computer with associated hard disk drive generates a prefetch table that contains pointers to disk

> locations and lengths of the records of an application program requested by the host computer during an initial power-on/reset."

Kroeker, Abstract.

> "The present invention relates generally to peripheral storage apparatus for computers, and more particularly to shortening the load time of computer programs from a hard disk drive to a host computer."

Kroeker, 1:9-12.

> "Accordingly, it is an object of the present invention to provide a method for rapidly communicating a computer program from a disk drive to a host computer."

Kroeker, 2:66-67.

1.1 maintaining a list of boot data used for booting a computer system;	Krocker, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Krocker discloses this limitation:

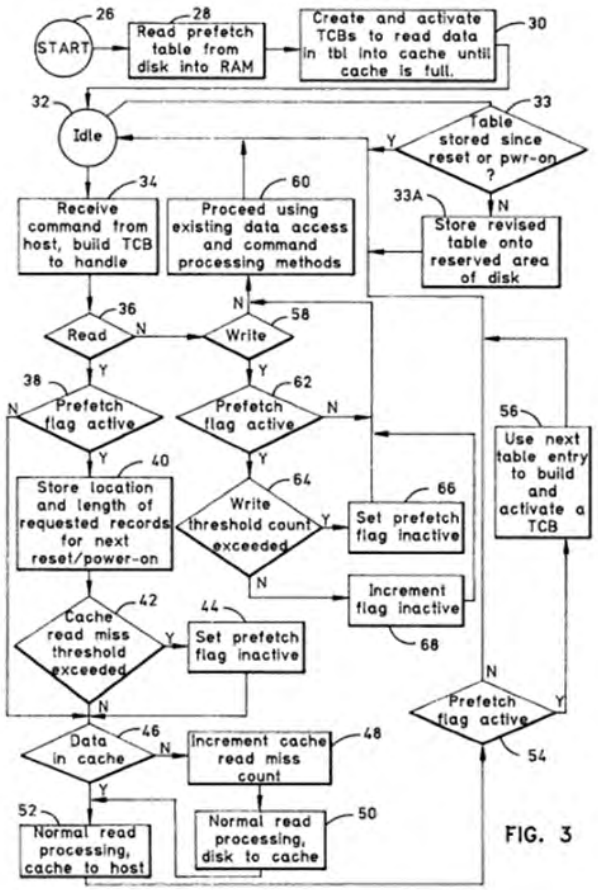


FIG. 3

Kroeker, Fig. 3.

> "A method for increasing boot speed of a host computer with associated hard disk drive generates a prefetch table that contains pointers to disk locations and lengths of the records of an application program requested

by the host computer during an initial power-on/reset."

Kroeker, Abstract.

> "In accordance with the present invention, steps executed by the digital processing apparatus include, after an initial power-up or reset of the hard disk drive and a host computer associated with the drive, receiving an initial read command from the host computer for transferring to the host computer a plurality of data records of a program stored on the disk. A prefetch table is then generated, with the table representing a disk location and length of each data record requested by the initial read command."

Kroeker, 2:20-37.

> "When the command is a read command, code means generate a prefetch table representative of at least the disk location of the records requested by the read command for transfer of the records from the disk to the cache for a subsequent power-on or reset of the host computer. Moreover, code means are provided for determining whether the records have been stored in the cache in response to a previous power-on or reset of the host computer, and the records are communicated to the host computer in response."

Kroeker, 3:21-29.

> "As discussed further below, the prefetch table contains a listing of the disk locations and lengths of data records that were requested by the host computer 14 in the immediately previous power-on/reset. Additionally, a copy of a prefetch flag, if enabled by the user, is created and set active at block 28."

Kroeker, 3:3-9. *See also* Kroeker, 3:9-16.

1.2 initializing a central processing unit of the computer system;	Krocker, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Krocker discloses this limitation:

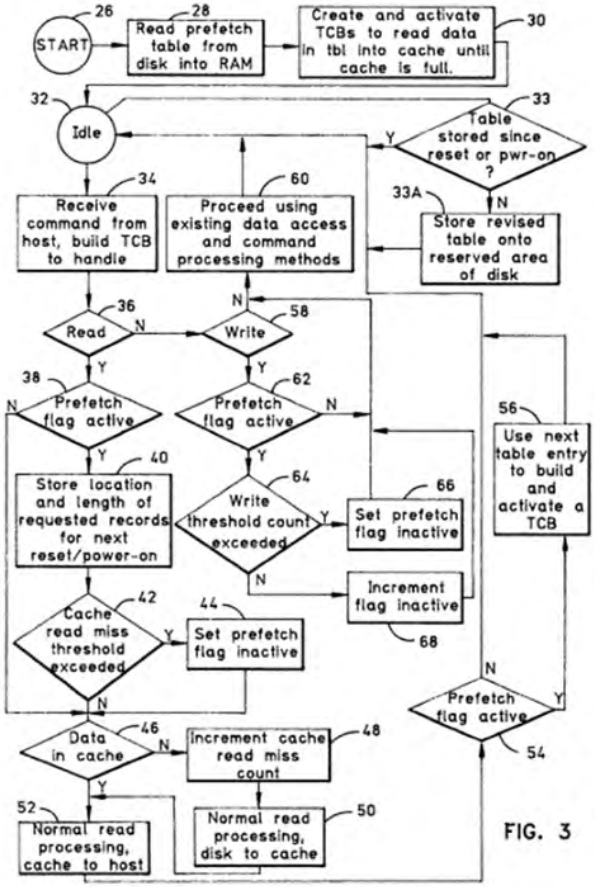


FIG. 3

Kroeker, Fig. 3.

> "During the next power-on/reset, before the host computer is ready for data but after the disk drive has completed its reset routine, using the prefetch table the disk drive accesses the previously requested data and copies it onto the cache of the disk drive, from where it is transferred to

> the host computer when the host computer requests it."
>
> Kroeker, Abstract.
>
> "Then, after a subsequent power-on or reset of the hard disk drive, and during a second power-on or reset of the host computer, the prefetch table is accessed to read into the data cache the data records."
>
> Kroeker, 2:37-40.
>
> "As disclosed in detail below, these code means are for enhanced loading of the program from the hard disk drive to the host computer during power-on or reset of the host computer. In accordance with the present invention, code means receive a command from the host computer during a power-up or reset of the host computer."
>
> Kroeker, 3:15-20.
>
> "Commencing at start state 26, the process moves to block 28, wherein a prefetch table is read from a reserved area of the disks 16 into the RAM cache 18."
>
> Kroeker, 5:1-3.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Krocker, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Krocker discloses this limitation:

> "During the next power-on/reset, before the host computer is ready for data but after the disk drive has completed its reset routine, using the prefetch table the disk drive accesses the previously requested data and copies it onto the cache of the disk drive, from where it is transferred to the host computer when the host computer requests it."

Kroeker, Abstract.

> "This invention is realized in a critical machine component that causes a digital processing apparatus to adaptively store a computer program on the cache of the hard disk drive and communicate the program to the host computer."

Kroeker, 2:23-26.

> "Then, after a subsequent power-on or reset of the hard disk drive, and during a second power-on or reset of the host computer, the prefetch table is accessed to read into the data cache the data records. In response to a subsequent read command from the host computer, it is determined whether records requested by the subsequent read command are stored in the data cache. If they are, the records are communicated from the cache to the host computer; otherwise, the records are communicated from the disk to the host computer."

Kroeker, 2:37-46.

> "In still another aspect, a computer hard disk drive includes at least one data storage disk and a data storage cache. Furthermore, the hard disk drive includes means for recording onto the cache, immediately after a hardware reset of the hard disk drive, data on the disk that has been requested by a host computer during a first hardware reset of the host computer."

Kroeker, 3:30-36.

> "Additionally, as intended by the present invention the onboard controller 20 includes an adaptive cache module 24. Per the present invention, the adaptive cache module 24 is executed by the onboard controller 20 as a series of computer-executable instructions."

Kroeker, 4:16-20.

> "From block 44, or from decision diamonds 38 or 42 when the decisions there are negative, the process moves to decision diamond 46 to determine whether the requested data exists in cache."

Kroeker, 5:65-6:1.

> "From block 50, or from decision diamond 46 if it was determined that the requested data exists in cache, the process moves to block 52 to transfer the record from cache 18 to the host computer 14."

Kroeker, 5:65-6:1.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Krocker, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Krocker discloses this limitation:

> "Then, after a subsequent power-on or reset of the hard disk drive, and during a second power-on or reset of the host computer, the prefetch table is accessed to read into the data cache the data records. In response to a subsequent read command from the host computer, it is determined whether records requested by the subsequent read command are stored in the data cache. If they are, the records are communicated from the cache to the host computer; otherwise, the records are communicated from the disk to the host computer."

Kroeker, 2:37-46.

> "Preferably, the accessing and determining steps are repeated for each power-on or reset of the host computer."

Kroeker, 2:47-48.

> "Additionally, the disk drive includes means for communicating the data from the cache to the host computer during a second hardware reset of

the host computer."

Kroeker, 3:36-38.

> "Additionally, as intended by the present invention the onboard controller 20 includes an adaptive cache module 24. Per the present invention, the adaptive cache module 24 is executed by the onboard controller 20 as a series of computer-executable instructions."

Kroeker, 4:16-20.

> "If it is active, the logic, at block 56, uses the next entry in the prefetch table to build a task control block (TCB) to fetch data into the same segment of the cache 18 that the just-transferred record had occupied prior to being communicated to the host computer 14. In accordance with the present invention, the TCB in block 50 is activated as though a command otherwise was received across the device/file interface. In other words, when the host computer 14 is a PC, the TCB in block 50 is activated as though a command otherwise was received across the SCSI (or IDE)--disk drive interface. In this way, the relatively small amount of cache storage space can be optimally used during the adaptive caching process until all records designated in the prefetch table have been loaded into cache and then transferred to the host computer 14."

Kroeker, 6:16-31.

> "If the command is not a write command, the process moves to block 60 to proceed using existing data access and command processing methods, and then the process continues back to the idle state 32."

Kroeker, 6:44-48.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Krocker, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Krocker discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

> "A method for increasing boot speed of a host computer with associated hard disk drive generates a prefetch table that contains pointers to disk locations and lengths of the records of an application program requested by the host computer during an initial power-on/reset."

Kroeker, Abstract.

> "As recognized by the present invention, however, it is possible to provide, without operating system intervention, a method for adaptively preparing a disk drive to effect rapid application program loading to a host computer."

Kroeker, 1:55-58.

> "And, the hard disk drive 12 can be any hard disk drive suitable for computer applications, provided that the hard disk drive 12 includes at least one, and typically a plurality of, data storage disks 16 and an on-board, solid state, random access memory (RAM) data cache 18."

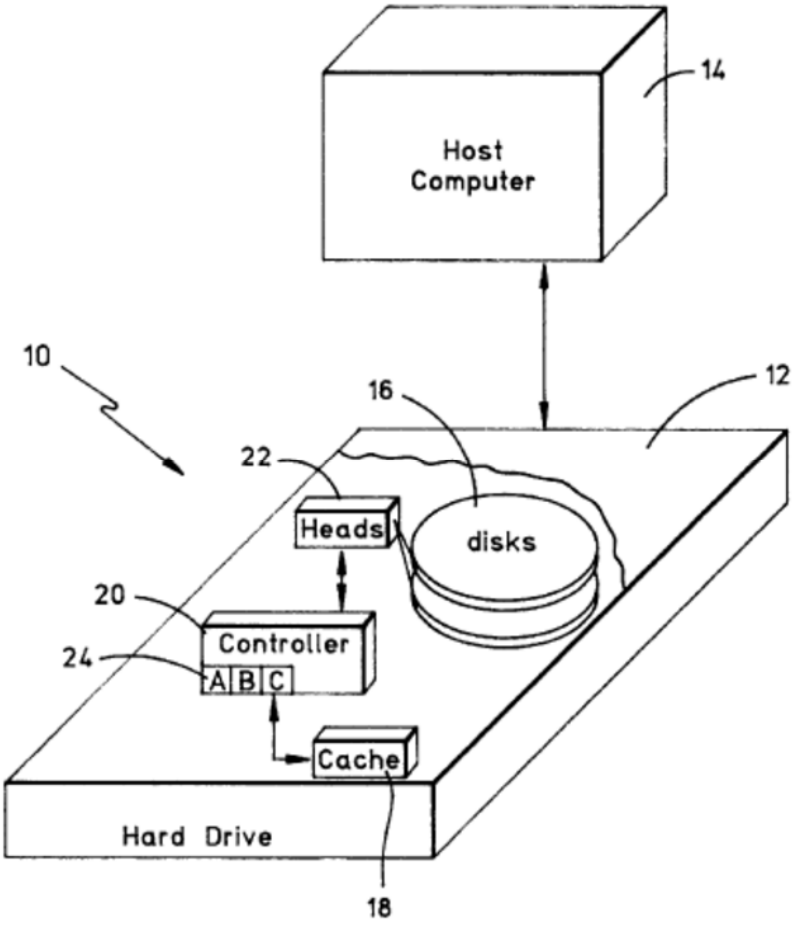
Kroeker, 4:5:10.

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Krocker, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Krocker discloses this limitation:

See Claims 1.3, and 1.4 above.



Kroeker, Fig. 1.

> "Indeed, the host computer 14 can be an embedded controller that is part of a music synthesizer, or part of an industrial instrument."

Kroeker, 4:3-6.

> "As shown in FIG. 1, the hard disk drive 12 also includes an onboard controller 20. In accordance with principles well-known in the art, the onboard controller 20 is a digital processor Which, among other things, controls read heads 22 in the disk drive 12 for effecting data transfer to and from the disks 16."

Kroeker, 4:11-16.

> "Additionally, as intended by the present invention the onboard controller 20 includes an adaptive cache module 24. Per the present invention, the adaptive cache module 24 is executed by the onboard controller 20 as a series of computer-executable instructions. These instructions are embodied as microcode in a memory, e.g., read-only memory (ROM) of the onboard controller 20. Such a ROM is indicated by reference numeral 21 in FIG. 2."

Kroeker, 4:17-24.

> "Manifestly, the invention may be practiced in its essential embodiment by a machine component, embodied by the ROM 21,that renders the computer program code elements in a form that instructs a digital processing apparatus (e.g., the onboard controller 20) to perform a sequence of function steps corresponding to those shown in the Figures. The machine component is shown in FIGS. 1 and 2 as a combination of program code elements A—C in computer readable form that are embodied in a computer usable data medium (the ROM 21) of the onboard controller 20."

Kroeker, 4:43-53, Fig. 3.

4. The method of claim 1, further comprising updating the list of boot data.	Krocker, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Krocker discloses this limitation:

See Claim 1.1 above.

> "The prefetch table is updated to reflect disk location changes for the various records, or to reflect new records that were requested by the host computer but not found in cache during the previous power-on/reset."

Kroeker, Abstract.

> "In accordance with the present invention, steps executed by the digital processing apparatus include, after an initial power-up or reset of the hard disk drive and a host computer associated with the drive, receiving an initial read command from the host computer for transferring to the host computer a plurality of data records of a program stored on the disk. A prefetch table is then generated, with the table representing a disk location and length of each data record requested by the initial read command."

Kroeker, 2:20-37.

> "Additionally, the steps further include updating the data prefetch table, communicating the records from the disk to the host computer when the read counter exceeds a predetermined threshold, and setting a prefetch flag to inactive when the read counter exceeds the predetermined threshold."

Kroeker, 2:59-64.

> "When the command is a read command, code means generate a prefetch table representative of at least the disk location of the records requested by the read command for transfer of the records from the disk to the cache for a subsequent power-on or reset of the host computer. Moreover, code means are provided for determining whether the records have been stored in the cache in response to a previous power-on or reset of the host computer, and the records are communicated to the host computer in

response."

Kroeker, 3:21-29.

> "As discussed further below, the prefetch table contains a listing of the disk locations and lengths of data records that were requested by the host computer 14 in the immediately previous power-on/reset. Additionally, a copy of a prefetch flag, if enabled by the user, is created and set active at block 28."

Kroeker, 3:3-9. *See also* Kroeker, 3:9-16.

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Krocker, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Krocker, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 1.1 and 4 above.	

Appendix A11
Invalidity of U.S. Patent 7,181,608 based on Krocker

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Krocker, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 1 (Preamble) above.	

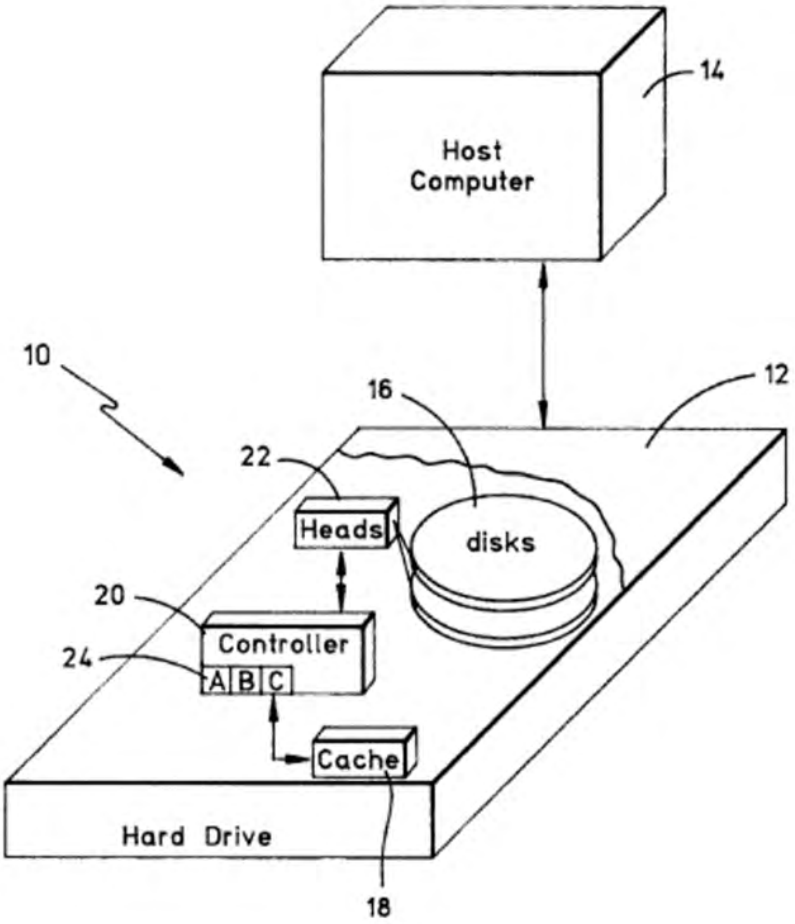
7.1 a digital signal processor (DSP) or controller;	Krocker, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Krocker discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also ***Look for additional references to controller or data storage controller***



Kroeker, Fig. 1.

> "Indeed, the host computer 14 can be an embedded controller that is part of a music synthesizer, or part of an industrial instrument."

Kroeker, 4:3-6.

> "As shown in FIG. 1, the hard disk drive 12 also includes an onboard controller 20. In accordance with principles well-known in the art, the onboard controller 20 is a digital processor Which, among other things, controls read heads 22 in the disk drive 12 for effecting data transfer to and from the disks 16."

Kroeker, 4:11-16.

> "Additionally, as intended by the present invention the onboard controller 20 includes an adaptive cache module 24. Per the present invention, the adaptive cache module 24 is executed by the onboard controller 20 as a series of computer-executable instructions. These instructions are embodied as microcode in a memory, e.g., read-only memory (ROM) of the onboard controller 20. Such a ROM is indicated by reference numeral 21 in FIG. 2."

Kroeker, 4:17-24.

> "Manifestly, the invention may be practiced in its essential embodiment by a machine component, embodied by the ROM 21,that renders the computer program code elements in a form that instructs a digital processing apparatus (e.g., the onboard controller 20) to perform a sequence of function steps corresponding to those shown in the Figures. The machine component is shown in FIGS. 1 and 2 as a combination of program code elements A—C in computer readable form that are embodied in a computer usable data medium (the ROM 21) of the onboard controller 20."

Kroeker, 4:43-53, Fig. 3.

7.2 a cache memory device; and;	Krocker, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Krocker, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Krocker, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Krocker, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Krocker, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Krocker discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> "A method for increasing boot speed of a host computer with associated hard disk drive generates a prefetch table that contains pointers to disk locations and lengths of the records of an application program requested by the host computer during an initial power-on/reset."

Kroeker, Abstract.

> "As recognized by the present invention, however, it is possible to provide, without operating system intervention, a method for adaptively preparing a disk drive to effect rapid application program loading to a host computer."

Kroeker, 1:55-58.

"And, the hard disk drive 12 can be any hard disk drive suitable for computer applications, provided that the hard disk drive 12 includes at least one, and typically a plurality of, data storage disks 16 and an on-board, solid state, random access memory (RAM) data cache 18." Kroeker, 4:5:10.

9.1 maintaining a list of application data associated with an application program;	Krocker, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Krocker, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Krocker, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Krocker discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> "A method for increasing boot speed of a host computer with associated hard disk drive generates a prefetch table that contains pointers to disk locations and lengths of the records of an application program requested by the host computer during an initial power-on/reset."

Kroeker, Abstract.

> "As recognized by the present invention, however, it is possible to provide, without operating system intervention, a method for adaptively preparing a disk drive to effect rapid application program loading to a host computer."

Kroeker, 1:55-58.

> "And, the hard disk drive 12 can be any hard disk drive suitable for computer applications, provided that the hard disk drive 12 includes at

least one, and typically a plurality of, data storage disks 16 and an on-board, solid state, random access memory (RAM) data cache 18." Kroeker, 4:5:10.

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Krocker, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Krocker discloses this limitation:

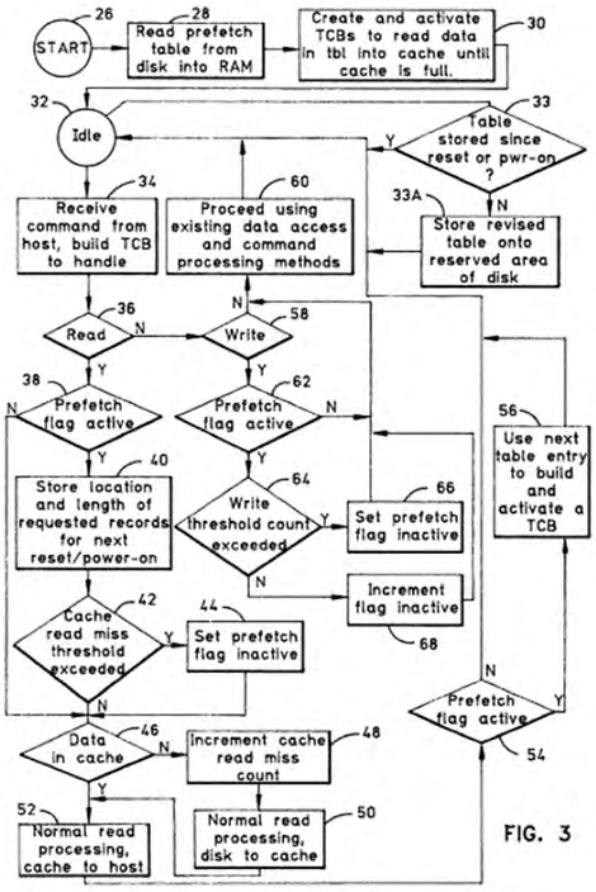


FIG. 3

Kroeker, Fig. 3.

> "A method for increasing boot speed of a host computer with associated hard disk drive generates a prefetch table that contains pointers to disk locations and lengths of the records of an application program requested by the host computer during an initial power-on/reset."

Kroeker, Abstract.

> "The present invention relates generally to peripheral storage apparatus for computers, and more particularly to shortening the load time of computer programs from a hard disk drive to a host computer."

Kroeker, 1:9-12.

> "In accordance with the present invention, steps executed by the digital processing apparatus include, after an initial power-up or reset of the hard disk drive and a host computer associated with the drive, receiving an initial read command from the host computer for transferring to the host computer a plurality of data records of a program stored on the disk. A prefetch table is then generated, with the table representing a disk location and length of each data record requested by the initial read command."

Kroeker, 2:20-37.

> "When the command is a read command, code means generate a prefetch table representative of at least the disk location of the records requested by the read command for transfer of the records from the disk to the cache for a subsequent power-on or reset of the host computer. Moreover, code means are provided for determining whether the records have been stored in the cache in response to a previous power-on or reset of the host computer, and the records are communicated to the host computer in response."

Kroeker, 3:21-29.

> "As discussed further below, the prefetch table contains a listing of the disk locations and lengths of data records that were requested by the host computer 14 in the immediately previous power-on/reset. Additionally, a copy of a prefetch flag, if enabled by the user, is created and set active at block 28."

Kroeker, 3:3-9. *See also* Kroeker, 3:9-16.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Krocker, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Krocker, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Krocker, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Krocker, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Krocker, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 1.3, 1.4, and 15 above.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Krocker, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Krocker, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above.	

22.1 maintaining a list of boot data used for booting a computer system;.	Krocker, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claim 1.1 above.	

22.2 initializing a central processing unit of the computer system;.	Krocker, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claim 1.2 above.	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Krocker, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claim 1.3 above.	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Krocker, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claim 1.4 above.	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Krocker, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 4, 10 and 11 above.	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Krocker, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 15 and 22 above.	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Krocker, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 16 and 22 above.	

27.1 a boot device..	Krocker, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claim 1 above.	

27.2 a processor..	Krocker, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claim 1.2 above.	

27.3 cache memory; and.	Krocker, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 1.3 and 1.4 above.	

27.4 non-volatile memory for storing logic code for use by the processor,..	Krocker, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claim 1.1 and 1.3 above.	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Krocker, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claim 1.1 above.	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Krocker, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claim 1.3 above.	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Krocker, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claim 1.4 above.	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Krocker, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 4, 10, and 11 above.	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Krocker, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 15 and 27 above.	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Krocker, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Krocker discloses this limitation: *See* Claims 16 and 27 above.	

Appendix A12
Invalidity of U.S. Patent 7,181,608 based on Lee

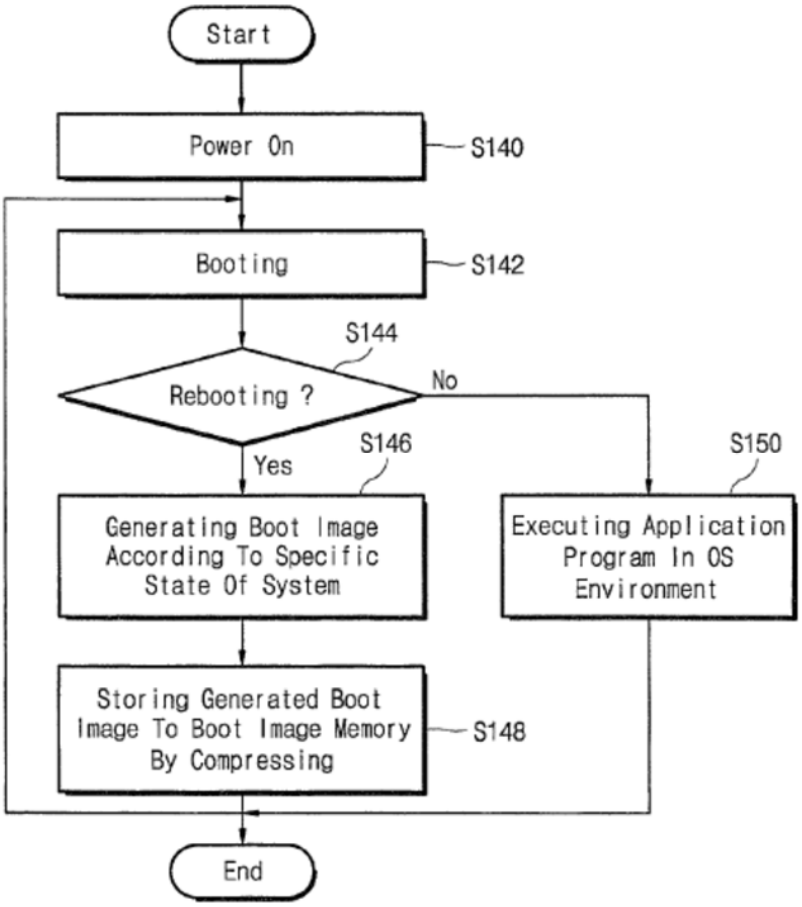
U.S. Patent Application Publication No. 2001/0039612 Lee ("Lee") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

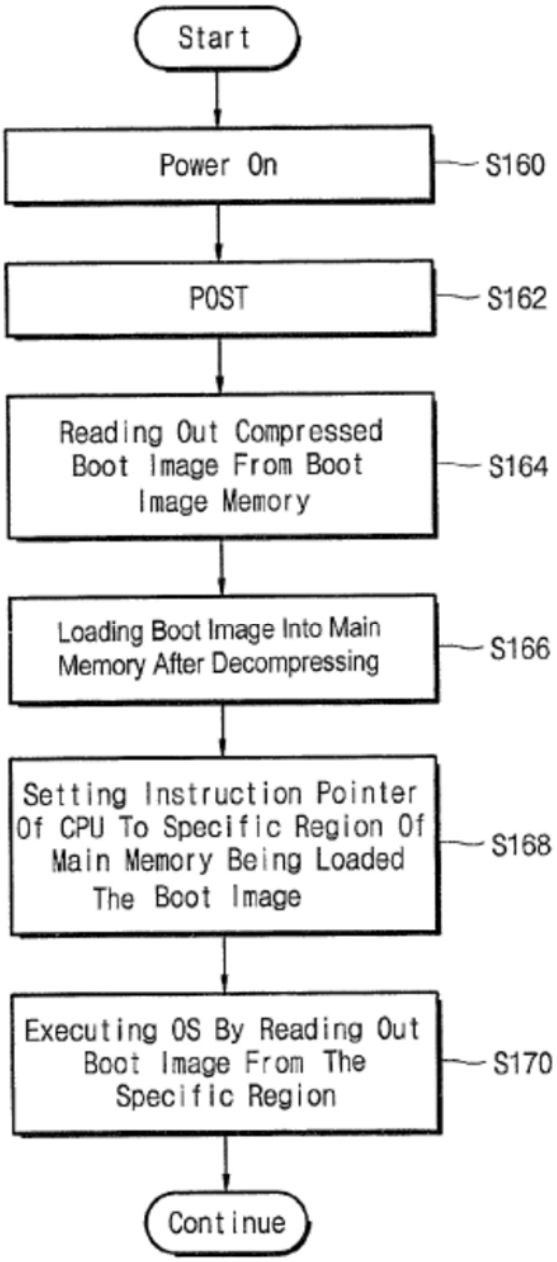
1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Lee, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:



Lee, Fig.2.



Lee, Fig. 3.

> "Assuming that the correct first diskette is inserted into the system, the ROM-based code retrieves the master boot record from sector 0 of the first diskette."

Lee, ¶ 0014.

> "The ROM-based code then copies the OS loader into RAM from the first

> diskette, starting at the address indicated in the master boot record and continuing for a length indicated by the offset provided by the master boot record. After copying the OS loader into RAM, the ROM-based code jumps to the OS loader."

Lee, ¶ 0014.

> "The OS loader is more sophisticated than the ROM-based code and performs certain preliminary functions, such as sizing memory. After performing preliminary functions, the OS loader copies into RAM a portion of the operating system known as the "kernel.""

Lee, ¶ 0015.

> "It is therefore an object of the present invention to provide a computer system to reduce booting time."

Lee, ¶ 0023.

> It is another object of the invention to provide a method for shutting off and booting a computer system to reduce booting time."

Lee, ¶ 0024.

> "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104."

Lee, ¶ 0040.

> "In other words, at the step S142, the CPU 102 executes the operating system if a successful boot is detected through a POST routine. Therefore, a certain application program can be executed in the operating system environment."

Lee, ¶ 0042.

1.1 maintaining a list of boot data used for booting a computer system;	Lee, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

> "The ROM-based code attempts to establish communication with a so-called "boot device". A boot device holds information that is necessary to boot the system. In attempting to establish communication with a boot device, the ROM-based code operates according to a so-called "boot order.""

Lee, ¶ 0012.

> "More particularly, the ROM-based code attempts to retrieve a so-called "master boot record" from a particular sector of the diskette. If the communication attempt is successful, the ROM-based code uses that device as the boot device. If not, the ROM-based code proceeds to attempt communication With a device of the next boot order, e.g., local media."

Lee, ¶ 0012.

> "In order to attain the above objects, according to an aspect of the present invention, there is provided a computer system having a central processing unit; a main and/or auxiliary power supply for supplying main and/or auxiliary power of the computer system; a boot image storing device for storing a boot image of the computer system; a main memory for storing the boot image from the boot image storing device by receiving the auxiliary power when the main power is shut off; and a composition memory for setting an instruction pointer of the central processing unit to a specific region of the main memory storing the boot image; wherein the central processing unit loads the boot image from the specific region of the main memory in response to the instruction pointer, thereby an operating system program can perform control functions."

Lee, ¶ 0025. *See also* Lee, ¶ 0026.

> "The boot image memory 108 is capable of containing a non-volatile memory such as a flash memory to store a compressed boot image data. The boot image data can be obtained by compressing an initial storing state of the main memory 104 as a data format."

Lee, ¶ 0038.

> "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104."

Lee, ¶ 0040.

> "FIG. 5 is a flow chart for illustrating a method for booting the computer system 200 shown in FIG. 4. The control flow is a program stored in the BIOS ROM 210, and is executed by the CPU 202 according to processing steps of the BIOS."

Lee, ¶ 0050. *See also* Lee, ¶ 0053.

1.2 initializing a central processing unit of the computer system;	Lee, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

> "When power is applied to a computer system, a portion of the computer system typically called the "initialization hardware" electronically detects the "power-on" condition and, in response to such a detection, forces certain circuitry of the system to a known state. For example, the CPU typically includes an instruction pointer (IP), which holds a memory address from which the CPU fetches an instruction to be executed by the CPU. The initialization hardware typically electronically forces the IP to an initial address so that the CPU may begin fetching and executing instructions from this initial address. The ROM is prerecorded With computer instructions, referred to below as the "ROM-based code." As a result, shortly after power on, the CPU begins executing the ROM-based code."

Lee, ¶ 0011.

> "The BIOS ROM 106 controls POST routine, interrupt processing, and system environment setting, according to initializing steps of the computer system 100. Especially, the BIOS ROM 106 sets the instruction pointer (IP)."

Lee, ¶ 0039.

> "In other words, at the step S142, the CPU 102 executes the operating system if a successful boot is detected through a POST routine. Therefore, a certain application program can be executed in the operating system environment."

Lee, ¶ 0042.

> "Referring to FIG. 3, the computer system 100 is powered on in step

S160, and POST routine is performed in step S162." Lee, ¶ 0045. *See also* Lee, ¶ 0051.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Lee, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

> "The ROM-based code attempts to establish communication with a so-called "boot device". A boot device holds information that is necessary to boot the system. In attempting to establish communication with a boot device, the ROM-based code operates according to a so-called "boot order.""

Lee, ¶ 0012.

> "More particularly, the ROM-based code attempts to retrieve a so-called "master boot record" from a particular sector of the diskette. If the communication attempt is successful, the ROM-based code uses that device as the boot device. If not, the ROM-based code proceeds to attempt communication With a device of the next boot order, e.g., local media."

Lee, ¶ 0012.

> "Assuming that the correct first diskette is inserted into the system, the ROM-based code retrieves the master boot record from sector 0 of the first diskette."

Lee, ¶ 0014.

> "The ROM-based code then copies the OS loader into RAM from the first diskette, starting at the address indicated in the master boot record and

> continuing for a length indicated by the offset provided by the master boot record. After copying the OS loader into RAM, the ROM-based code jumps to the OS loader."

Lee, ¶ 0014.

> "The OS loader is more sophisticated than the ROM-based code and performs certain preliminary functions, such as sizing memory. After performing preliminary functions, the OS loader copies into RAM a portion of the operating system known as the "kernel.""

Lee, ¶ 0015.

> "In order to attain the above objects, according to an aspect of the present invention, there is provided a computer system having a central processing unit; a main and/or auxiliary power supply for supplying main and/or auxiliary power of the computer system; a boot image storing device for storing a boot image of the computer system; a main memory for storing the boot image from the boot image storing device by receiving the auxiliary power when the main power is shut off; and a composition memory for setting an instruction pointer of the central processing unit to a specific region of the main memory storing the boot image; wherein the central processing unit loads the boot image from the specific region of the main memory in response to the instruction pointer, thereby an operating system program can perform control functions."

Lee, ¶ 0025. *See also* Lee, ¶ 0026.

> "The boot image memory 108 is capable of containing a non-volatile memory such as a flash memory to store a compressed boot image data. The boot image data can be obtained by compressing an initial storing state of the main memory 104 as a data format."

Lee, ¶ 0038.

> "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104."

Lee, ¶ 0040.

> "Continually, at step S164, the compressed boot image is read out. At

> step S166, the compressed boot image is loaded to the main memory 104 after decompressing. At step S168, an instruction pointer (IP) of the CPU 102 is set to a specific region of the main memory 104 being loaded the boot image. And then at step S170, the operating system is executed by reading out the boot image from the specific region."

Lee, ¶ 0045. *See also* Lee, ¶ 0048.

> "Continually, at step S224, a compressed boot image 216 is loaded from the CD-ROM 214, and the boot image is loaded to the main memory 206 after decompressing in step S226. At step S228, an instruction pointer 204 of the CPU 202 is set to a specific region 208 of the main memory 206 being loaded the boot image. At step S230, the operating system is executed by reading out the boot image from the specific region 208 of the main memory 206. As a result, the operating system can perform control functions."

Lee, ¶ 0051.

> "The CPU 302 reads out the boot image 324 from the hard disk drive 320 and loads it to the main memory 304, under control of the BIOS. In other words, the CPU 302 decompresses the compressed boot image from the specific region of the hard disk drive 320, and loads it to a specific region of the main memory 304. The CPU 302 reads out the location information of the boot image from the BIOS ROM 306, and then reads out the boot image from the specific region of the main memory 304 according to the information. Thus, the operating system can perform control functions 15 by setting the IP of the CPU 302 to the specific region of the main memory 304."

Lee, ¶ 0054.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Lee, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

> "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104."

Lee, ¶ 0040.

> "Continually, at step S164, the compressed boot image is read out. At step S166, the compressed boot image is loaded to the main memory 104 after decompressing. At step S168, an instruction pointer (IP) of the CPU 102 is set to a specific region of the main memory 104 being loaded the boot image. And then at step S170, the operating system is executed by reading out the boot image from the specific region."

Lee, ¶ 0045. *See also* Lee, ¶ 0048.

> "Continually, at step S224, a compressed boot image 216 is loaded from

> the CD-ROM 214, and the boot image is loaded to the main memory 206 after decompressing in step S226. At step S228, an instruction pointer 204 of the CPU 202 is set to a specific region 208 of the main memory 206 being loaded the boot image. At step S230, the operating system is executed by reading out the boot image from the specific region 208 of the main memory 206. As a result, the operating system can perform control functions."
>
> Lee, ¶ 0051.
>
> "The CPU 302 reads out the boot image 324 from the hard disk drive 320 and loads it to the main memory 304, under control of the BIOS. In other words, the CPU 302 decompresses the compressed boot image from the specific region of the hard disk drive 320, and loads it to a specific region of the main memory 304. The CPU 302 reads out the location information of the boot image from the BIOS ROM 306, and then reads out the boot image from the specific region of the main memory 304 according to the information. Thus, the operating system can perform control functions 15 by setting the IP of the CPU 302 to the specific region of the main memory 304."
>
> Lee, ¶ 0054.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Lee, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "The initial storing state is capable of executing a certain application program in an operating system program environment."

Lee, ¶ 0038.

> "In other words, at the step S142, the CPU 102 executes the operating system if a successful boot is detected through a POST routine. Therefore, a certain application program can be executed in the operating system environment."

Lee, ¶ 0042.

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Lee, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

> "The computer system 100 is an IBM compatible computer system, Which includes a plurality of controllers(for example, an input/output (I/O) controller 110, a hard disk drive (HDD) controller 112, and a floppy disk drive(EDD) controller 114), input devices including a keyboard 118 and a mouse 120, and auxiliary storing devices including a hard disk drive (HDD) 122, a CD-ROM drive 124, a floppy disk drive (FDD) 126, and so on. Further, the computer system 100 includes a video controller 116, and a display 128."

Lee, ¶ 0037.

> "In addition, the computer system 300 further includes a hard disk controller 308, and a hard disk drive (HDD) 320 storing an operating system program 322 and boot image 324."

Lee, ¶ 0052.

4. The method of claim 1, further comprising updating the list of boot data.	Lee, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.1 above.	

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Lee, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Lee, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Lee, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Lee, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

> "The computer system 100 is an IBM compatible computer system, Which includes a plurality of controllers(for example, an input/output (I/O) controller 110, a hard disk drive (HDD) controller 112, and a floppy disk drive(EDD) controller 114), input devices including a keyboard 118 and a mouse 120, and auxiliary storing devices including a hard disk drive (HDD) 122, a CD-ROM drive 124, a floppy disk drive (FDD) 126, and so on. Further, the computer system 100 includes a video controller 116, and a display 128."

Lee, ¶ 0037.

> "In addition, the computer system 300 further includes a hard disk controller 308, and a hard disk drive (HDD) 320 storing an operating system program 322 and boot image 324."

Lee, ¶ 0052.

7.2 a cache memory device; and;	Lee, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Lee, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Lee, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Lee, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Lee, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also ***Look for additional references to application programs***

> "The initial storing state is capable of executing a certain application program in an operating system program environment."

Lee, ¶ 0038.

> "In other words, at the step S142, the CPU 102 executes the operating system if a successful boot is detected through a POST routine. Therefore, a certain application program can be executed in the operating system environment."

Lee, ¶ 0042.

9.1 maintaining a list of application data associated with an application program;	Lee, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Lee, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Lee, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

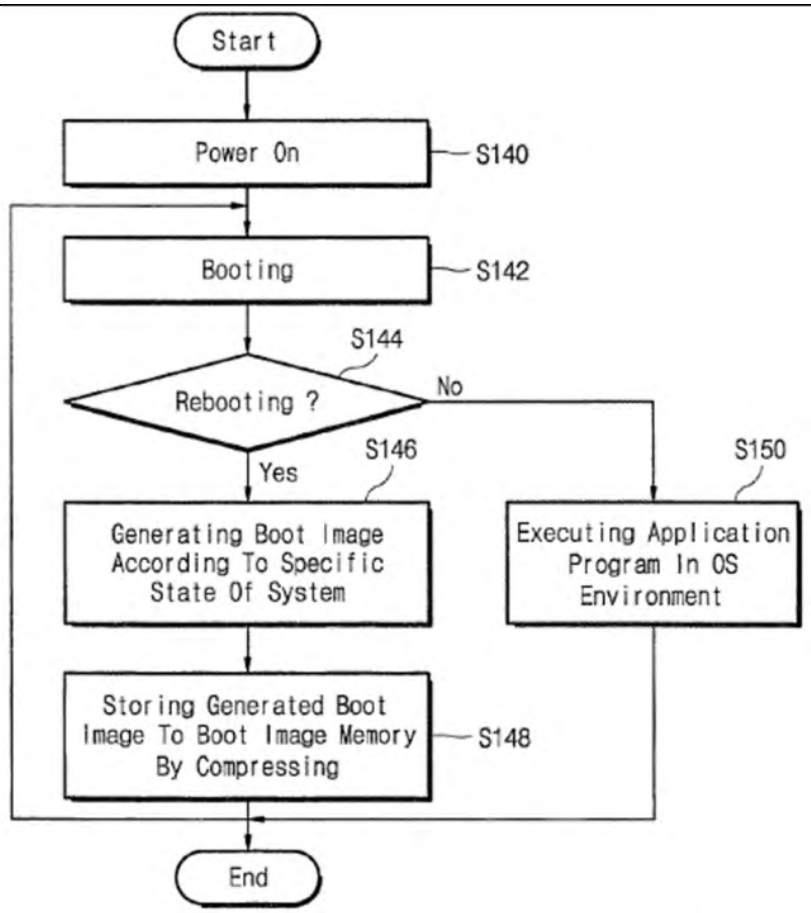
> "The initial storing state is capable of executing a certain application program in an operating system program environment."

Lee, ¶ 0038.

> "In other words, at the step S142, the CPU 102 executes the operating system if a successful boot is detected through a POST routine. Therefore, a certain application program can be executed in the operating system environment."

Lee, ¶ 0042.

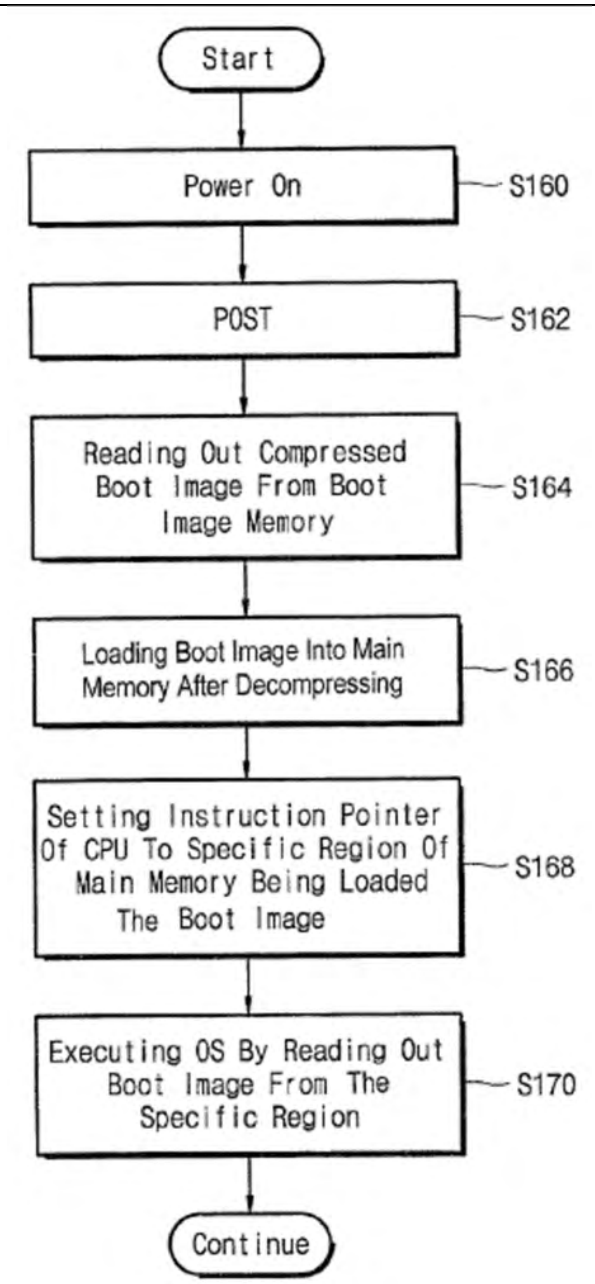
10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Lee, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* ***Copy from 936 element 1.1***	



Lee, Fig. 2.

Lee Claim 10

"The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device"



Lee, Fig. 3.

> "The ROM-based code attempts to establish communication with a so-called "boot device". A boot device holds information that is necessary to boot the system. In attempting to establish communication with a boot device, the ROM-based code operates according to a so-called "boot order.""

Lee, ¶ 0012.

> "More particularly, the ROM-based code attempts to retrieve a so-called "master boot record" from a particular sector of the diskette. If the communication attempt is successful, the ROM-based code uses that device as the boot device. If not, the ROM-based code proceeds to attempt communication With a device of the next boot order, e.g., local media."

Lee, ¶ 0012.

> "In order to attain the above objects, according to an aspect of the present invention, there is provided a computer system having a central processing unit; a main and/or auxiliary power supply for supplying main and/or auxiliary power of the computer system; a boot image storing device for storing a boot image of the computer system; a main memory for storing the boot image from the boot image storing device by receiving the auxiliary power when the main power is shut off; and a composition memory for setting an instruction pointer of the central processing unit to a specific region of the main memory storing the boot image; wherein the central processing unit loads the boot image from the specific region of the main memory in response to the instruction pointer, thereby an operating system program can perform control functions."

Lee, ¶ 0025. *See also* Lee, ¶ 0026.

> "The boot image memory 108 is capable of containing a non-volatile memory such as a flash memory to store a compressed boot image data. The boot image data can be obtained by compressing an initial storing state of the main memory 104 as a data format."

Lee, ¶ 0038.

> "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104."

Lee, ¶ 0040.

> "In other words, at the step S142, the CPU 102 executes the operating system if a successful boot is detected through a POST routine. Therefore, a certain application program can be executed in the operating system

environment."

Lee, ¶ 0042.

> "At step S148, the generated boot image is stored to the boot image memory 108 after compressing, and then the computer system 100 is rebooted."

Lee, ¶ 0043.

> "FIG. 5 is a flow chart for illustrating a method for booting the computer system 200 shown in FIG. 4. The control flow is a program stored in the BIOS ROM 210, and is executed by the CPU 202 according to processing steps of the BIOS."

Lee, ¶ 0050. *See also* Lee, ¶ 0053.

> "Continually, at step S224, a compressed boot image 216 is loaded from the CD-ROM 214, and the boot image is loaded to the main memory 206 after decompressing in step S226. At step S228, an instruction pointer 204 of the CPU 202 is set to a specific region 208 of the main memory 206 being loaded the boot image. At step S230, the operating system is executed by reading out the boot image from the specific region 208 of the main memory 206. As a result, the operating system can perform control functions."

Lee, ¶ 0051.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Lee, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

> "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104."

Lee, ¶ 0040.

> "Continually, at step S164, the compressed boot image is read out. At step S166, the compressed boot image is loaded to the main memory 104 after decompressing. At step S168, an instruction pointer (IP) of the CPU 102 is set to a specific region of the main memory 104 being loaded the boot image. And then at step S170, the operating system is executed by reading out the boot image from the specific region."

Lee, ¶ 0045. *See also* Lee, ¶ 0048.

> "Continually, at step S224, a compressed boot image 216 is loaded from the CD-ROM 214, and the boot image is loaded to the main memory 206 after decompressing in step S226. At step S228, an instruction pointer 204 of the CPU 202 is set to a specific region 208 of the main memory 206 being loaded the boot image. At step S230, the operating system is executed by reading out the boot image from the specific region 208 of the main memory 206. As a result, the operating system can perform control functions."

Lee, ¶ 0051.

> "The CPU 302 reads out the boot image 324 from the hard disk drive 320 and loads it to the main memory 304, under control of the BIOS. In other words, the CPU 302 decompresses the compressed boot image from the

specific region of the hard disk drive 320, and loads it to a specific region of the main memory 304. The CPU 302 reads out the location information of the boot image from the BIOS ROM 306, and then reads out the boot image from the specific region of the main memory 304 according to the information. Thus, the operating system can perform control functions 15 by setting the IP of the CPU 302 to the specific region of the main memory 304."

Lee, ¶ 0054.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Lee, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Lee, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Lee, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Lee, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

See also

> "At step S148, the generated boot image is stored to the boot image memory 108 after compressing, and then the computer system 100 is rebooted."

Lee, ¶ 0043.

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Lee, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Lee, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above.	

22.1 maintaining a list of boot data used for booting a computer system;.	Lee, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.1 above.	

22.2 initializing a central processing unit of the computer system;.	Lee, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.2 above.	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Lee, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.3 above.	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Lee, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.4 above.	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Lee, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 4, 10 and 11 above.	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Lee, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 15 and 22 above.	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Lee, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 16 and 22 above.	

27.1 a boot device..	Lee, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1 above.	

27.2 a processor..	Lee, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.2 above.	

27.3 cache memory; and.	Lee, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.3 and 1.4 above.	

27.4 non-volatile memory for storing logic code for use by the processor,..	Lee, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.1 and 1.3 above.	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Lee, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.1 above.	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Lee, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.3 above.	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Lee, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.4 above.	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Lee, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 4, 10, and 11 above.	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Lee, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 15 and 27 above.	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Lee, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 16 and 27 above.	

Appendix A13
Invalidity of U.S. Patent 7,181,608 based on Makinen

U.S. Patent No. 6,237,080 to Makinen ("Makinen") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A13
Invalidity of U.S. Patent 7,181,608 based on Makinen

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Makinen, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> According to a first aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) with a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), the method comprising reading a set of compressed RISC operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions.

Makinen, 1:63-2:4

> According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU)

and a read only memory (ROM), the method comprising reading a set of compressed operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.

The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.

Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

> According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only memory (ROM), and a set of compressed operating instructions stored in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions, and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM.
>
> Makinen, 2:60-3:3:2

1.1 maintaining a list of boot data used for booting a computer system;	Makinen, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> According to a first aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) with a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), the method comprising reading a set of compressed RISC operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions.

Makinen, 1:63-2:4

> According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) and a read only memory (ROM), the method comprising reading a set of

compressed operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.

The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.

Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only

> memory (ROM), and a set of compressed operating instructions stored in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions, and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM.

Makinen, 2:60-3:3:2

> The flash ROM 4 is used to store a set of RISC operating instructions which define the basic input/output system (BIOS) of the microprocessor as well as the device drivers, libraries, and user applications. The instructions are in compressed form, having previously been compressed using the Pkzip compression program

Makinen, 3:45-50

> It will be appreciated that the operating system of the computer may be altered by directly accessing the flash ROM 5 to erase and/or rewrite compressed instructions 6 stored therein. However, in some circumstances it may be desirable for the end-user to be able to alter the compressed operating instructions 6, or indeed for the computer itself to be able to 'dynamically' alter the instructions. To this end, the Pkzip compression method may also be stored in the flash ROM 4.

Makinen, 4:10-17

1.2 initializing a central processing unit of the computer system;	Makinen, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications.

Makinen, 1:10-19

> Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed

> instruction set 7, from the ROM 4. Thereafter the microprocessor is directed, by the decompressed set, to read and decompress the compressed instruction set 6. The expanded instruction set 6′ is then stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen, 3:66-4:9

See also Makinen, Fig. 3

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Makinen, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the

microprocessor together with device drivers, libraries, and user applications.

Makinen, 1:10-19

According to a first aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) with a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), the method comprising reading a set of compressed RISC operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions.

Makinen, 1:63-2:4

According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) and a read only memory (ROM), the method comprising reading a set of compressed operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.

The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.

Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this

> case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

> According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

> According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only memory (ROM), and a set of compressed operating instructions stored in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions, and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM.

Makinen, 2:60-3:3:2

> The flash ROM 4 is used to store a set of RISC operating instructions which define the basic input/output system (BIOS) of the microprocessor as well as the device drivers, libraries, and user applications. The instructions are in compressed form, having previously been compressed using the Pkzip compression program

Makinen, 3:45-50

> In order to enable the microprocessor 1 to decompress the stored compressed instructions, the flash ROM 4 additionally stores a set of instructions, in uncompressed form, which define the Pkzip decompression program. FIG. 2 shows a memory map of the ROM 4 prior to booting the computer, where a part of the memory space is occupied by the compressed instructions 6 and a part is occupied by the uncompressed Pkzip instructions 7. In FIG. 2, the ROM 4 is shown

> coupled to the microprocessor 1, as is the RAM 3 which at this stage remains empty.

Makinen, 3:56-65

> Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed instruction set 7, from the ROM 4. Thereafter the microprocessor is directed, by the decompressed set, to read and decompress the compressed instruction set 6. The expanded instruction set 6′ is then stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen, 3:66-4:9

> In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6,8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of the operating instructions 6′,8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen, 4:18-32

See also Makinen, Figs. 2-4

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Makinen, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications.

Makinen, 1:10-19

> According to a first aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) with a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), the method comprising reading a set of compressed RISC operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions.

Makinen, 1:63-2:4

> According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) and a read only memory (ROM), the method comprising reading a set of compressed operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.
>
> The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.
>
> Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

> Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

> According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

> According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only memory (ROM), and a set of compressed operating instructions stored in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions, and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM.

Makinen, 2:60-3:3:2

> The flash ROM 4 is used to store a set of RISC operating instructions which define the basic input/output system (BIOS) of the microprocessor as well as the device drivers, libraries, and user applications. The instructions are in compressed form, having previously been compressed using the Pkzip compression program

Makinen, 3:45-50

> In order to enable the microprocessor 1 to decompress the stored compressed instructions, the flash ROM 4 additionally stores a set of instructions, in uncompressed form, which define the Pkzip decompression program. FIG. 2 shows a memory map of the ROM 4 prior to booting the computer, where a part of the memory space is occupied by the compressed instructions 6 and a part is occupied by the uncompressed Pkzip instructions 7. In FIG. 2, the ROM 4 is shown coupled to the microprocessor 1, as is the RAM 3 which at this stage remains empty.

Makinen, 3:56-65

> Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed instruction set 7, from the ROM 4. Thereafter the microprocessor is directed, by the decompressed set, to read and decompress the compressed instruction set 6. The expanded instruction set 6′ is then stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen, 3:66-4:9

> In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6, 8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of the operating instructions 6′, 8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen, 4:18-32

See also Makinen, Figs. 2-4

Appendix A13
Invalidity of U.S. Patent 7,181,608 based on Makinen

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Makinen, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

> At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications.

Makinen, 1:10-19

> The flash ROM 4 is used to store a set of RISC operating instructions which define the basic input/output system (BIOS) of the microprocessor as well as the device drivers, libraries, and user applications. The instructions are in compressed form, having previously been compressed using the Pkzip compression program. The compressed instructions occupy considerably less flash ROM space than would the corresponding uncompressed instructions, e.g. ½ to ⅕ of the memory space. Such a high compression ratio results from the particular structure of RISC instructions.

Makinen, 3:46-55

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Makinen, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claims 1.3, and 1.4 above.

> Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed instruction set 7, from the ROM 4. Thereafter the microprocessor is directed, by the decompressed set, to read and decompress the compressed instruction set 6. The expanded instruction set 6′ is then stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen 3:66-4:28

> In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6,8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of the operating instructions 6′, 8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen 4:18-37

Appendix A13
Invalidity of U.S. Patent 7,181,608 based on Makinen

4. The method of claim 1, further comprising updating the list of boot data.	Makinen, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claim 1.1 above.

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications.

Makinen, 1:10-19

> According to a first aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) with a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), the method comprising reading a set of compressed RISC operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions.

Makinen, 1:63-2:4

> According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) and a read only memory (ROM), the method comprising reading a set of compressed operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.
>
> The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.
>
> Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

> Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

> According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

> According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only memory (ROM), and a set of compressed operating instructions stored in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions, and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM.

Makinen, 2:60-3:3:2

> It will be appreciated that the operating system of the computer may be altered by directly accessing the flash ROM 5 to erase and/or rewrite compressed instructions 6 stored therein. However, in some circumstances it may be desirable for the end-user to be able to alter the compressed operating instructions 6, or indeed for the computer itself to be able to 'dynamically' alter the instructions. To this end, the Pkzip compression method may also be stored in the flash ROM 4.

Makinen, 4:10-17

> In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6,8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended

version of the operating instructions 6′, 8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen, 4:18-32

See also Makinen, Figs. 2-4

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Makinen, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Makinen, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Makinen, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Makinen, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

> Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed instruction set 7, from the ROM 4. Thereafter the microprocessor is directed, by the decompressed set, to read and decompress the compressed instruction set 6. The expanded instruction set 6′ is then stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen 3:66-4:28

> In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6,8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of the operating instructions 6′,8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen 4:18-37

7.2 a cache memory device; and;	Makinen, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Makinen, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Makinen, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Makinen, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Makinen, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications.

Makinen, 1:10-19

The flash ROM 4 is used to store a set of RISC operating instructions which define the basic input/output system (BIOS) of the microprocessor as well as the device drivers, libraries, and user applications. The instructions are in compressed form, having previously been compressed using the Pkzip compression program. The compressed

instructions occupy considerably less flash ROM space than would the corresponding uncompressed instructions, e.g. ½ to ⅕ of the memory space. Such a high compression ratio results from the particular structure of RISC instructions. Makinen, 3:46-55

9.1 maintaining a list of application data associated with an application program;	Makinen, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Makinen, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Makinen, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claims 1.4, 2, and 8 above.

At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications.

Makinen, 1:10-19

The flash ROM 4 is used to store a set of RISC operating instructions which define the basic input/output system (BIOS) of the microprocessor as well as the device drivers, libraries, and user applications. The instructions are in compressed form, having previously been compressed using the Pkzip compression program. The compressed instructions occupy considerably less flash ROM space than would the corresponding

uncompressed instructions, e.g. ½ to ⅕ of the memory space. Such a high compression ratio results from the particular structure of RISC instructions. Makinen, 3:46-55

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Makinen, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications.

Makinen, 1:10-19

> According to a first aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) with a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), the method comprising reading a set of compressed RISC operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions.

Makinen, 1:63-2:4

> According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) and a read only memory (ROM), the method comprising reading a set of compressed operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.
>
> The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.
>
> Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

> Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

> According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

> According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only memory (ROM), and a set of compressed operating instructions stored in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions, and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM.

Makinen, 2:60-3:3:2

> The flash ROM 4 is used to store a set of RISC operating instructions which define the basic input/output system (BIOS) of the microprocessor as well as the device drivers, libraries, and user applications. The instructions are in compressed form, having previously been compressed using the Pkzip compression program

Makinen, 3:45-50

> In order to enable the microprocessor 1 to decompress the stored compressed instructions, the flash ROM 4 additionally stores a set of instructions, in uncompressed form, which define the Pkzip decompression program. FIG. 2 shows a memory map of the ROM 4 prior to booting the computer, where a part of the memory space is occupied by the compressed instructions 6 and a part is occupied by the uncompressed Pkzip instructions 7. In FIG. 2, the ROM 4 is shown coupled to the microprocessor 1, as is the RAM 3 which at this stage remains empty.

Makinen, 3:56-65

> Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed instruction set 7, from the ROM 4. Thereafter the microprocessor is directed, by the decompressed set, to read and decompress the compressed instruction set 6. The expanded instruction set 6′ is then stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen, 3:66-4:9

> It will be appreciated that the operating system of the computer may be altered by directly accessing the flash ROM 5 to erase and/or rewrite compressed instructions 6 stored therein. However, in some circumstances it may be desirable for the end-user to be able to alter the compressed operating instructions 6, or indeed for the computer itself to be able to 'dynamically' alter the instructions. To this end, the Pkzip compression method may also be stored in the flash ROM 4.

Makinen, 4:10-17

> In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6,8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of the operating instructions 6′,8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen, 4:18-32

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Makinen, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications.

Makinen, 1:10-19

> According to a first aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) with a reduced instruction set computer (RISC) architecture, and a read

only memory (ROM), the method comprising reading a set of compressed RISC operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions.

Makinen, 1:63-2:4

According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) and a read only memory (ROM), the method comprising reading a set of compressed operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.

The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.

Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction

> set computer (RISC) architecture, and a read only memory (ROM), there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

> According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only memory (ROM), and a set of compressed operating instructions stored in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions, and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM.

Makinen, 2:60-3:3:2

> The flash ROM 4 is used to store a set of RISC operating instructions which define the basic input/output system (BIOS) of the microprocessor as well as the device drivers, libraries, and user applications. The instructions are in compressed form, having previously been compressed using the Pkzip compression program

Makinen, 3:45-50

> In order to enable the microprocessor 1 to decompress the stored compressed instructions, the flash ROM 4 additionally stores a set of instructions, in uncompressed form, which define the Pkzip decompression program. FIG. 2 shows a memory map of the ROM 4 prior to booting the computer, where a part of the memory space is occupied by the compressed instructions 6 and a part is occupied by the uncompressed Pkzip instructions 7. In FIG. 2, the ROM 4 is shown coupled to the microprocessor 1, as is the RAM 3 which at this stage remains empty.

Makinen, 3:56-65

> Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed instruction set 7, from the ROM 4. Thereafter the microprocessor is directed, by the decompressed set, to read and decompress the

> compressed instruction set 6. The expanded instruction set 6′ is then stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen, 3:66-4:9

> In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6,8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of the operating instructions 6′,8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen, 4:18-32

See also Makinen, Figs. 2-4

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Makinen, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Makinen, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claims 1.3 and 1.4 above.

It will be appreciated that the operating system of the computer may be altered by directly accessing the flash ROM 5 to erase and/or rewrite compressed instructions 6 stored therein. However, in some circumstances it may be desirable for the end-user to be able to alter the compressed operating instructions 6, or indeed for the computer itself to be able to 'dynamically' alter the instructions. To this end, the Pkzip compression method may also be stored in the flash ROM 4.

Makinen, 4:9-17

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Makinen, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claims 1.3 and 1.4 above.

The flash ROM 4 is used to store a set of RISC operating instructions which define the basic input/output system (BIOS) of the microprocessor as well as the device drivers, libraries, and user applications. The instructions are in compressed form, having previously been compressed using the Pkzip compression program. The compressed instructions occupy considerably less flash ROM space than would the corresponding uncompressed instructions, e.g. ½ to ⅕ of the memory space. Such a high compression ratio results from the particular structure of RISC instructions.

Makinen, 3:46-55

It will be appreciated by the person of skill in the art that other modifications may be made to the embodiments described above without departing from the scope of the present invention. For example, compression algorithms other than Pkzip may be used, including Gzip-9, Zoo-a, and Arj-a. Compression methods may also be optimised for ARM (Trade Mark) processors.

Makinen, 4:38-45

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Makinen, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Makinen, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Makinen, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Makinen, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Makinen, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Makinen, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Makinen, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Makinen, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Makinen, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Makinen, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Makinen, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Makinen, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Makinen, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Makinen, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Makinen, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Makinen, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Makinen, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Makinen, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Makinen, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 15 and 27 above	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Makinen, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 16 and 27 above	

Appendix A14
Invalidity of U.S. Patent 7,181,608 based on Noll

U.S. Patent No. 5,793,943 to Noll ("Noll") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A14
Invalidity of U.S. Patent 7,181,608 based on Noll

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Noll, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Noll discloses this limitation:

> "A computer system includes a dual basic input-output system (BIOS) read-only memory (ROM) system to initialize the computer. When the computer is first powered on or reset, the primary BIOS ROM is initially enabled. The computer analyzes the entire program contents of the primary BIOS memory to detect data errors. If a data error is detected, a chip enable circuit disables the primary BIOS ROM and enables a secondary BIOS ROM containing essentially the same initialization instructions as the primary BIOS ROM. If no errors are detected in the secondary BIOS ROM, the initialization of the computer proceeds using the secondary BIOS ROM. As part of the initialization procedure, the contents of the secondary BIOS ROM are copied to a random access memory. The primary BIOS ROM can then be reprogrammed with the contents of the secondary BIOS ROM using the copy in random access memory, or from the secondary BIOS ROM itself."

Noll, Abstract.

> "For example, when the computer is first powered up or reset, a software program, typically designated as a "basic input-output system" (BIOS) initializes the computer and permits the startup of an operating system, such as Microsoft MS-DOS®. The BIOS program typically resides in a read-only memory (ROM). If the BIOS ROM is defective for any reason, the computer will not function properly. Therefore, it can be appreciated that there is a significant need for a system to recover from a BIOS ROM failure in a manner that does not require user intervention. The present invention provides this and other advantages as will be apparent from the

following detailed description and accompanying figures."

Noll, 1:29-42.

> "The present invention is embodied in a system for the automatic recovery of a BIOS ROM failure. In one embodiment, the system includes a first BIOS memory that contains a series of computer instructions to initialize the computer. The first BIOS memory has a chip enable input that is initially enabled. An error detection circuit analyzes data contained within the first BIOS memory and detects errors therein. The error detection circuit generates an error signal upon detection of errors in the first BIOS memory. The system also includes a second BIOS memory containing the series of computer instructions to initialize the computer and also having a chip enable input. An enabling circuit is included to disable the first BIOS memory chip enable input and to enable the second BIOS memory chip enable input in response to the error signal. This effectively causes the computer system to switch to the second BIOS memory so that the series of computer instructions to initialize the computer are executed from the second BIOS memory rather the first BIOS memory."

Noll, 1:44-63.

> "Another reason that the primary BIOS ROM 22 is copied into the RAM 14 is that some data in the primary BIOS ROM may be in a compressed format and must be decompressed before the CPU 12 can use it."

Noll, 4:66-5:2.

1.1 maintaining a list of boot data used for booting a computer system;	Noll, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Noll discloses this limitation:

> "A computer system includes a dual basic input-output system (BIOS) read-only memory (ROM) system to initialize the computer. When the computer is first powered on or reset, the primary BIOS ROM is initially enabled. The computer analyzes the entire program contents of the primary BIOS memory to detect data errors. If a data error is detected, a chip enable circuit disables the primary BIOS ROM and enables a secondary BIOS ROM containing essentially the same initialization instructions as the primary BIOS ROM. If no errors are detected in the secondary BIOS ROM, the initialization of the computer proceeds using the secondary BIOS ROM. As part of the initialization procedure, the contents of the secondary BIOS ROM are copied to a random access memory. The primary BIOS ROM can then be reprogrammed with the contents of the secondary BIOS ROM using the copy in random access memory, or from the secondary BIOS ROM itself."

Noll, Abstract.

> "The present invention is embodied in a system for the automatic recovery of a BIOS ROM failure. In one embodiment, the system includes a first BIOS memory that contains a series of computer instructions to initialize the computer. The first BIOS memory has a chip enable input that is initially enabled. An error detection circuit analyzes data contained within the first BIOS memory and detects errors therein. The error detection circuit generates an error signal upon detection of errors in the first BIOS memory. The system also includes a second BIOS memory containing the series of computer instructions to initialize the computer and also having a chip enable input. An enabling circuit is included to disable the first BIOS memory chip enable input and to enable the second BIOS memory chip enable input in response to the error signal. This effectively causes the computer system to switch to the second BIOS memory so that the series of computer instructions to

initialize the computer are executed from the second BIOS memory rather the first BIOS memory."

Noll, 1:44-63.

1.2 initializing a central processing unit of the computer system;	Noll, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Noll discloses this limitation:

> "A computer system includes a dual basic input-output system (BIOS) read-only memory (ROM) system to initialize the computer. When the computer is first powered on or reset, the primary BIOS ROM is initially enabled. The computer analyzes the entire program contents of the primary BIOS memory to detect data errors. If a data error is detected, a chip enable circuit disables the primary BIOS ROM and enables a secondary BIOS ROM containing essentially the same initialization instructions as the primary BIOS ROM. If no errors are detected in the secondary BIOS ROM, the initialization of the computer proceeds using the secondary BIOS ROM. As part of the initialization procedure, the contents of the secondary BIOS ROM are copied to a random access memory. The primary BIOS ROM can then be reprogrammed with the contents of the secondary BIOS ROM using the copy in random access memory, or from the secondary BIOS ROM itself."

Noll, Abstract.

> "For example, when the computer is first powered up or reset, a software program, typically designated as a "basic input-output system" (BIOS) initializes the computer and permits the startup of an operating system, such as Microsoft MS-DOS®. The BIOS program typically resides in a read-only memory (ROM). If the BIOS ROM is defective for any reason, the computer will not function properly. Therefore, it can be appreciated that there is a significant need for a system to recover from a BIOS ROM failure in a manner that does not require user intervention. The present invention provides this and other advantages as will be apparent from the following detailed description and accompanying figures."

Noll, 1:29-42.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Noll, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Noll discloses this limitation:

> "A computer system includes a dual basic input-output system (BIOS) read-only memory (ROM) system to initialize the computer. When the computer is first powered on or reset, the primary BIOS ROM is initially enabled. The computer analyzes the entire program contents of the primary BIOS memory to detect data errors. If a data error is detected, a chip enable circuit disables the primary BIOS ROM and enables a secondary BIOS ROM containing essentially the same initialization instructions as the primary BIOS ROM. If no errors are detected in the secondary BIOS ROM, the initialization of the computer proceeds using the secondary BIOS ROM. As part of the initialization procedure, the contents of the secondary BIOS ROM are copied to a random access memory. The primary BIOS ROM can then be reprogrammed with the contents of the secondary BIOS ROM using the copy in random access memory, or from the secondary BIOS ROM itself."

Noll, Abstract.

> "The present invention is embodied in a system for the automatic recovery of a BIOS ROM failure. In one embodiment, the system includes a first BIOS memory that contains a series of computer instructions to initialize the computer. The first BIOS memory has a chip enable input that is initially enabled. An error detection circuit analyzes data contained within the first BIOS memory and detects errors therein. The error detection circuit generates an error signal upon detection of errors in the first BIOS memory. The system also includes a second BIOS

memory containing the series of computer instructions to initialize the computer and also having a chip enable input. An enabling circuit is included to disable the first BIOS memory chip enable input and to enable the second BIOS memory chip enable input in response to the error signal. This effectively causes the computer system to switch to the second BIOS memory so that the series of computer instructions to initialize the computer are executed from the second BIOS memory rather the first BIOS memory."

Noll, 1:44-63.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Noll, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Noll discloses this limitation:

> "A computer system includes a dual basic input-output system (BIOS) read-only memory (ROM) system to initialize the computer. When the computer is first powered on or reset, the primary BIOS ROM is initially enabled. The computer analyzes the entire program contents of the primary BIOS memory to detect data errors. If a data error is detected, a chip enable circuit disables the primary BIOS ROM and enables a secondary BIOS ROM containing essentially the same initialization instructions as the primary BIOS ROM. If no errors are detected in the secondary BIOS ROM, the initialization of the computer proceeds using the secondary BIOS ROM. As part of the initialization procedure, the contents of the secondary BIOS ROM are copied to a random access memory. The primary BIOS ROM can then be reprogrammed with the contents of the secondary BIOS ROM using the copy in random access memory, or from the secondary BIOS ROM itself."

Noll, Abstract.

> "For example, when the computer is first powered up or reset, a software program, typically designated as a "basic input-output system" (BIOS)

> initializes the computer and permits the startup of an operating system, such as Microsoft MS-DOS®. The BIOS program typically resides in a read-only memory (ROM). If the BIOS ROM is defective for any reason, the computer will not function properly. Therefore, it can be appreciated that there is a significant need for a system to recover from a BIOS ROM failure in a manner that does not require user intervention. The present invention provides this and other advantages as will be apparent from the following detailed description and accompanying figures."

Noll, 1:29-42.

> "Another reason that the primary BIOS ROM 22 is copied into the RAM 14 is that some data in the primary BIOS ROM may be in a compressed format and must be decompressed before the CPU 12 can use it."

Noll, 4:66-5:2.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Noll, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Noll, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Noll discloses this limitation:

See Claims 1.3, and 1.4 above.

> The computer 10 also includes a chip enable circuit 36 that selectively enables and disables the primary BIOS ROM 22 and the secondary BIOS ROM 30. When the computer 10 is first powered up, the chip enable circuit 36 activates a ROMSEL1 control line 40 and a ROMSEL2 control line 42 by setting both of these control lines to a high logic level. The ROMSEL1 control line 40 and the ROMSEL2 control line 42 serve as inputs to an AND gate 44 whose output is coupled to the chip enable input 24 on the primary BIOS ROM 22. Thus, the chip enable circuit 36 initially activates the chip enable input 24 on the primary BIOS ROM 22.

Noll, 3:25-37

> The ROMSEL2 control line 42 is also coupled to the input of an inverter 48 whose output is coupled to the input of an AND gate 50. The ROMSEL1 control line 40 serves as another input to the AND gate 50. The output of the AND gate 50 is coupled to the chip enable input 32 of the secondary BIOS ROM 30. The inversion of the ROMSEL2 control line 42 assures that the secondary BIOS ROM 30 is initially disabled when the computer 10 is first powered up, or reset. In the event that the computer 10 detects an error in the primary BIOS ROM 22, the chip enable circuit 36 causes the ROMSEL2 control line 42 to change to a low logic level. This disables the primary BIOS ROM 22, while also enabling the secondary BIOS ROM 30.

Noll, 3:38-50

> The chip enable circuit 36 may be virtually any type of data storage element, such as a register, that can be controlled by the CPU 12. In the presently preferred embodiment, the chip enable circuit 36 is part of a peripheral

component interconnect (PCI) local bus to an industry standard architecture (ISA) bus bridge chip (not shown). The computer 10 switches between the primary BIOS ROM 22 and the secondary BIOS ROM 30 by toggling the ROMSEL to control line 42 in the PCI to ISA bridge chip (not shown). However, those of ordinary skill in the art will readily recognize that any programmable register will operate satisfactorily with the computer 10.

Noll, 3:50-61

4. The method of claim 1, further comprising updating the list of boot data.	Noll, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claim 1.1 above.	

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Noll, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Noll, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1.1 and 4 above.	

Appendix A14
Invalidity of U.S. Patent 7,181,608 based on Noll

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Noll, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Noll, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Noll discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

> The computer 10 also includes a chip enable circuit 36 that selectively enables and disables the primary BIOS ROM 22 and the secondary BIOS ROM 30. When the computer 10 is first powered up, the chip enable circuit 36 activates a ROMSEL1 control line 40 and a ROMSEL2 control line 42 by setting both of these control lines to a high logic level. The ROMSEL1 control line 40 and the ROMSEL2 control line 42 serve as inputs to an AND gate 44 whose output is coupled to the chip enable input 24 on the primary BIOS ROM 22. Thus, the chip enable circuit 36 initially activates the chip enable input 24 on the primary BIOS ROM 22.

Noll, 3:25-37

> The ROMSEL2 control line 42 is also coupled to the input of an inverter 48 whose output is coupled to the input of an AND gate 50. The ROMSEL1 control line 40 serves as another input to the AND gate 50. The output of the AND gate 50 is coupled to the chip enable input 32 of the secondary BIOS ROM 30. The inversion of the ROMSEL2 control line 42 assures that the secondary BIOS ROM 30 is initially disabled when the computer 10 is first powered up, or reset. In the event that the computer 10 detects an error in the primary BIOS ROM 22, the chip enable circuit 36 causes the ROMSEL2 control line 42 to change to a low logic level. This disables the primary BIOS ROM 22, while also enabling the secondary BIOS ROM 30.

Noll, 3:38-50

> The chip enable circuit 36 may be virtually any type of data storage element, such as a register, that can be controlled by the CPU 12. In the presently preferred embodiment, the chip enable circuit 36 is part of a peripheral component interconnect (PCI) local bus to an industry standard architecture (ISA) bus bridge chip (not shown). The computer 10 switches between the primary BIOS ROM

22 and the secondary BIOS ROM 30 by toggling the ROMSEL to control line 42 in the PCI to ISA bridge chip (not shown). However, those of ordinary skill in the art will readily recognize that any programmable register will operate satisfactorily with the computer 10.

Noll, 3:50-61

7.2 a cache memory device; and;	Noll, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Noll, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Noll, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Noll, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Noll, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.	

9.1 maintaining a list of application data associated with an application program;	Noll, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Noll, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Noll, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1.4, 2, and 8 above.	

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Noll, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Noll discloses this limitation:

> "A computer system includes a dual basic input-output system (BIOS) read-only memory (ROM) system to initialize the computer. When the computer is first powered on or reset, the primary BIOS ROM is initially enabled. The computer analyzes the entire program contents of the primary BIOS memory to detect data errors. If a data error is detected, a chip enable circuit disables the primary BIOS ROM and enables a secondary BIOS ROM containing essentially the same initialization instructions as the primary BIOS ROM. If no errors are detected in the secondary BIOS ROM, the initialization of the computer proceeds using the secondary BIOS ROM. As part of the initialization procedure, the contents of the secondary BIOS ROM are copied to a random access memory. The primary BIOS ROM can then be reprogrammed with the contents of the secondary BIOS ROM using the copy in random access memory, or from the secondary BIOS ROM itself."

Noll, Abstract.

> "The present invention is embodied in a system for the automatic recovery of a BIOS ROM failure. In one embodiment, the system includes a first BIOS memory that contains a series of computer instructions to initialize the computer. The first BIOS memory has a chip enable input that is initially enabled. An error detection circuit analyzes data contained within the first BIOS memory and detects errors therein. The error detection circuit generates an error signal upon detection of errors in the first BIOS memory. The system also includes a second BIOS memory containing the series of computer instructions to initialize the computer and also having a chip enable input. An enabling circuit is

included to disable the first BIOS memory chip enable input and to enable the second BIOS memory chip enable input in response to the error signal. This effectively causes the computer system to switch to the second BIOS memory so that the series of computer instructions to initialize the computer are executed from the second BIOS memory rather the first BIOS memory."

Noll, 1:44-63.

"Another reason that the primary BIOS ROM 22 is copied into the RAM 14 is that some data in the primary BIOS ROM may be in a compressed format and must be decompressed before the CPU 12 can use it."

Noll, 4:66-5:2.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Noll, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Noll discloses this limitation:

> "A computer system includes a dual basic input-output system (BIOS) read-only memory (ROM) system to initialize the computer. When the computer is first powered on or reset, the primary BIOS ROM is initially enabled. The computer analyzes the entire program contents of the primary BIOS memory to detect data errors. If a data error is detected, a chip enable circuit disables the primary BIOS ROM and enables a secondary BIOS ROM containing essentially the same initialization instructions as the primary BIOS ROM. If no errors are detected in the secondary BIOS ROM, the initialization of the computer proceeds using the secondary BIOS ROM. As part of the initialization procedure, the contents of the secondary BIOS ROM are copied to a random access memory. The primary BIOS ROM can then be reprogrammed with the contents of the secondary BIOS ROM using the copy in random access memory, or from the secondary BIOS ROM itself."

Noll, Abstract.

> "For example, when the computer is first powered up or reset, a software program, typically designated as a "basic input-output system" (BIOS) initializes the computer and permits the startup of an operating system, such as Microsoft MS-DOS®. The BIOS program typically resides in a read-only memory (ROM). If the BIOS ROM is defective for any reason, the computer will not function properly. Therefore, it can be appreciated that there is a significant need for a system to recover from a BIOS ROM failure in a manner that does not require user intervention. The present invention provides this and other advantages as will be apparent from the following detailed description and accompanying figures."

Noll, 1:29-42.

> "Another reason that the primary BIOS ROM 22 is copied into the RAM 14 is that some data in the primary BIOS ROM may be in a

compressed format and must be decompressed before the CPU 12 can use it." Noll, 4:66-5:2.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Noll, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Noll, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Noll discloses this limitation:

See Claims 1.3 and 1.4 above.

> As part of the normal setup procedures, instructions contained within the primary BIOS ROM 22 cause the CPU 12 to perform a "Power-On Self Test" (POST). The POST procedure may include routines such as a check of the RAM 14, peripheral devices (not shown), and the like. The POST procedure is well known to those of ordinary skill in the art, and need not be described in detail herein. As part of the POST procedure, the data in the primary BIOS ROM 22 is copied to the RAM 14. This process is sometimes known as shadowing to reflect the fact that a complete copy of the data in the primary BIOS ROM 22 is copied or shadowed into the RAM 14. One reason for copying the primary BIOS ROM 22 into the RAM 14 is that the access time for the RAM is typically much less than the access time for the primary BIOS ROM. Thus, the instructions contained within the primary BIOS ROM 22 are copied into the RAM 14 so that the instructions may be executed more rapidly by the CPU 12. Another reason that the primary BIOS ROM 22 is copied into the RAM 14 is that some data in the primary BIOS ROM may be in a compressed format and must be decompressed before the CPU 12 can use it. An exact copy of the data in the primary BIOS ROM 22 is initially copied into a low address portion of the RAM 14. After the primary BIOS ROM 22 has been copied into the low address portion of the RAM 14, the chip enable circuit 36 sets the ROMSEL1 control line 40 to a low level, thus disabling both the primary BIOS ROM 22 and the secondary BIOS ROM 30. The CPU 12 subsequently executes instructions from the low address portion of the RAM 14. A different portion (not shown) of the RAM 14, typically having an address space that overlap with the address space of the primary BIOS ROM 22, is designated as a "shadow RAM." Once the CPU 12 is executing instructions from the low address portion of the RAM 14, the data in the low address portion is decompressed if necessary and copied to the shadow RAM (not shown). At this point in time, the CPU 12 will begin executing instructions from the addresses in the RAM 14 that correspond with the addresses in the BIOS ROM 22 until the POST procedure is completed. The computer 10 can

use the shadow RAM address space because the primary BIOS ROM 22 and the secondary BIOS ROM 30 have both been disabled by the low logic level on the ROMSEL1 control line 40. At that point, the computer 10 will boot the operating system, such as MS-DOS®.

Noll, 4:49-5:24

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Noll, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Noll, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1.3, 1.4, and 15 above.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Noll, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Noll, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Noll, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Noll, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Noll, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Noll, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Noll, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Noll, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Noll, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Noll, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Noll, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Noll, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Noll, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Noll, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Noll, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Noll, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Noll, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Noll, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 15 and 27 above	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Noll, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Noll discloses this limitation: *See* Claims 16 and 27 above	

Appendix A15
Invalidity of U.S. Patent 7,181,608 based on Pearce

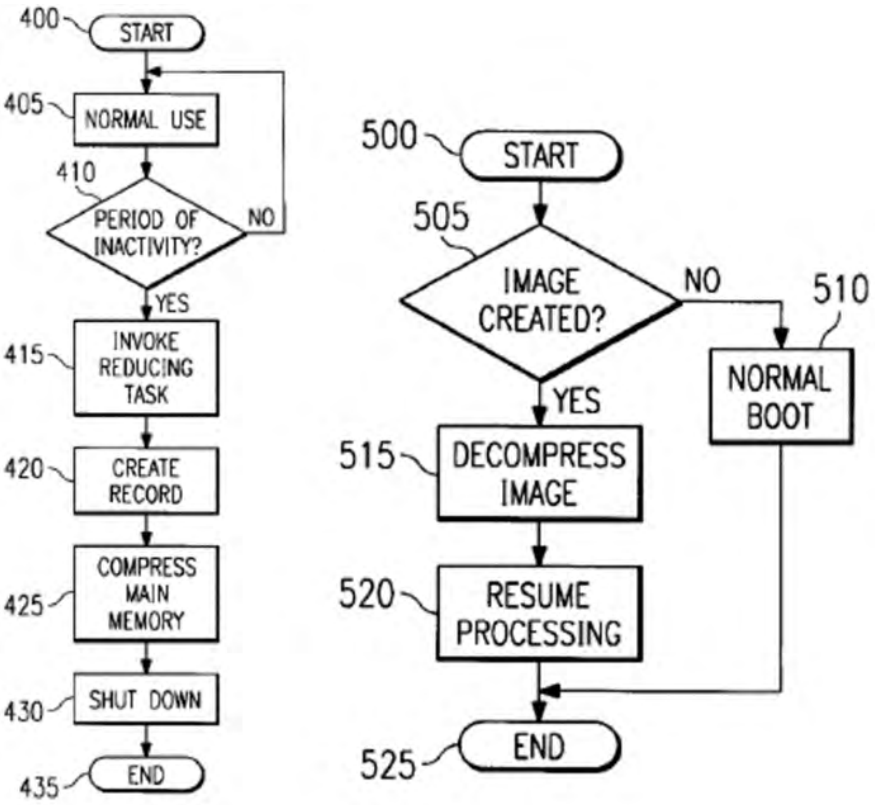
U.S. Patent No. 5,828,877 to Pearce ("Pearce") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Pearce, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Pearce discloses this limitation:



Pearce, Fig. 4, Fig. 5.

> "The most effective solution to power saving following a long period of user inactivity is to turn off the PC completely. However, it is unacceptable simply to interrupt has not yet been stored in non-volatile memory, particularly the hard disk drive. Rather, it is advantageous to store the contents of the volatile main memory of the PC as an image in

> the nonvolatile secondary storage device (a so-called "suspend-to-disk" or "STD"). When the user restores power to the PC, the image is restored to the main memory (a "resume-from-disk" or "RFD"), placing the PC in exactly the same functional state as it Was When power was interrupted."

Pearce, 1:65-2:10.

> "Execution begins in a start block 300 wherein the portable PC 100 is started (or "booted") and proceeds to a block 305 wherein the user engages the portable PC 100 in normal use. During normal use, an operating system is loaded into the main memory 210."

Pearce, 5:52-55, Fig. 3.

See also Pearce 4:16-22.

1.1 maintaining a list of boot data used for booting a computer system;	Pearce, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Pearce discloses this limitation:

> "During compression of the main memory, the record containing the identity of the allocable units compressed is also stored for use by the main memory restoration method detailed in FIG. 5."

Pearce, 8:27-30, Fig. 5.

> "As a part of the block 515, the units previously allocated to the reducing task are marked as unallocated and therefore free for subsequent allocation by the operating system."

Pearce, 8:67-9:3, Fig. 5.

1.2 initializing a central processing unit of the computer system;	Pearce, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Pearce discloses this limitation:

> "The most effective solution to power saving following a long period of user inactivity is to turn off the PC completely. However, it is unacceptable simply to interrupt has not yet been stored in non-volatile memory, particularly the hard disk drive. Rather, it is advantageous to store the contents of the volatile main memory of the PC as an image in the nonvolatile secondary storage device (a so-called "suspend-to-disk" or "STD"). When the user restores power to the PC, the image is restored to the main memory (a "resume-from-disk" or "RFD"), placing the PC in exactly the same functional state as it Was When power was interrupted."

Pearce, 1:65-2:10.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Pearce, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Pearce discloses this limitation:

> "The most effective solution to power saving following a long period of user inactivity is to turn off the PC completely. However, it is unacceptable simply to interrupt has not yet been stored in non-volatile memory, particularly the hard disk drive. Rather, it is advantageous to store the contents of the volatile main memory of the PC as an image in the nonvolatile secondary storage device (a so-called "suspend-to-disk" or "STD"). When the user restores power to the PC, the image is restored to the main memory (a "resume-from-disk" or "RFD"), placing the PC in exactly the same functional state as it was when power was interrupted."

Pearce, 1:65-2:10.

> "Accordingly, a first embodiment of the present invention provides, in a computer system having a CPU, a main memory (either real or virtual) divisible into allocable units, a secondary storage unit and an operating system for allocating the allocable units to tasks for use thereby, a suspend circuit for creating an optimized compressed image of data in the main memory."

Pearce, 2:36-43.

> "Once the contents of main memory have been compressed and stored as an image in the secondary storage device, power to the computer system is interrupted."

Pearce, 2:57-59.

> "The present invention is designed to conserve computer power by storing the pertinent contents of the main memory 210 as a compressed image in the nonvolatile secondary storage device 260, allowing power to be interrupted to the portable PC 100 in its entirety. When the user so directs, power is restored to the PC, and the image is decompressed back into the main memory 210, allowing the CPU 200 to continue processing at the point where it stopped prior to interruption of power."

Pearce, 5:36-44.

> "Execution begins in a start block 500 wherein the portable PC 100 boots, loading initial portions of code commonly stored in nonvolatile memory within the portable PC 100."

Pearce, 8:39-41, Fig. 5.

> "If a compressed image file is detected, execution proceeds to a block 515, wherein a corresponding (e.g., run length decoding) decompression routine decompresses and restores the main memory 210 to the state in which it was prior to shutdown. In the first embodiment, the units previously allocated to the reducing task remain filled with the bit pattern. In the second embodiment, the stored record is retrieved and used as a guide by the decompression routine to restore the compressed allocable units to their proper position in main memory."

Pearce, 8:52-61, Fig. 5.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Pearce, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Pearce discloses this limitation:

> "The most effective solution to power saving following a long period of user inactivity is to turn off the PC completely. However, it is unacceptable simply to interrupt has not yet been stored in non-volatile memory, particularly the hard disk drive. Rather, it is advantageous to store the contents of the volatile main memory of the PC as an image in the nonvolatile secondary storage device (a so-called "suspend-to-disk" or "STD"). When the user restores power to the PC, the image is restored to the main memory (a "resume-from-disk" or "RFD"), placing the PC in exactly the same functional state as it was when power was interrupted."

Pearce, 1:65-2:10.

> "Accordingly, a first embodiment of the present invention provides, in a computer system having a CPU, a main memory (either real or virtual) divisible into allocable units, a secondary storage unit and an operating system for allocating the allocable units to tasks for use thereby, a suspend circuit for creating an optimized compressed image of data in the main memory."

Pearce, 2:36-43.

> "In a preferred embodiment, the suspend circuit further comprises a circuit for restoring the main memory by decompressing the compressed image into the main memory. The allocable units allocated to the reducing task are designated as unallocated units in the decompressed image."

Pearce, 3:47-52.

> "The present invention is designed to conserve computer power by storing the pertinent contents of the main memory 210 as a compressed image in the nonvolatile secondary storage device 260, allowing power to be interrupted to the portable PC 100 in its entirety. When the user so directs, power is restored to the PC, and the image is decompressed back into the main memory 210, allowing the CPU 200 to continue processing at the point where it stopped prior to interruption of power."

Pearce, 5:36-44.

> "If a compressed image file is detected, execution proceeds to a block 515, Wherein a corresponding (e.g., run length decoding) decompression routine decompresses and restores the main memory 210 to the state in which it was prior to shutdown. In the first embodiment, the units previously allocated to the reducing task remain filled with the bit pattern. In the second embodiment, the stored record is retrieved and used as a guide by the decompression routine to restore the compressed allocable units to their proper position in main memory."

Pearce, 8:52-61, Fig. 5.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Pearce, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Pearce discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "One of the purposes of the operating system is to allocate resources to tasks (or "application programs," such as Word processors, spreadsheets, communications programs, database managers, games and the like and their associated data) as they are executed on the portable PC 100."

Pearce, 5:55-61.

> "The user again loads an application task and begins to edit a document. For purposes of this example, it is again assumed that the user leaves the portable PC 100 Without saving the document. A period of inactivity thus begins."

Pearce, 7:65-8:3.

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Pearce, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Pearce discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

> "A bus controller 250 manages communication of data between the relatively fast local bus 240 and a relatively slow expansion bus 290 via lines 251, 252, respectively."

Pearce, 5:17-19.

4. The method of claim 1, further comprising updating the list of boot data.	Pearce, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Pearce discloses this limitation:

See Claim 1.1 above.

See also

> "In a preferred embodiment, the suspend circuit further comprises a circuit for restoring the main memory by decompressing the compressed image into the main memory. The allocable units allocated to the reducing task are designated as unallocated units in the decompressed image."

Pearce, 3:47-52.

> "As a part of the block 515, the units previously allocated to the reducing task are marked as unallocated and therefore free for subsequent allocation by the operating system."

Pearce, 8:67-9:3, Fig. 5.

See also Pearce 4:16-22.

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Pearce, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Pearce, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 1.1 and 4 above.	

Appendix A15
Invalidity of U.S. Patent 7,181,608 based on Pearce

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Pearce, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Pearce, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Pearce discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

> "A bus controller 250 manages communication of data between the relatively fast local bus 240 and a relatively slow expansion bus 290 via lines 251, 252, respectively."

Pearce, 5:17-19.

7.2 a cache memory device; and;	Pearce, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Pearce, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Pearce, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Pearce, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Pearce, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Pearce discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> "One of the purposes of the operating system is to allocate resources to tasks (or "application programs," such as Word processors, spreadsheets, communications programs, database managers, games and the like and their associated data) as they are executed on the portable PC 100."

Pearce, 5:55-61.

> "The user again loads an application task and begins to edit a document. For purposes of this example, it is again assumed that the user leaves the portable PC 100 Without saving the document. A period of inactivity thus begins."

Pearce, 7:65-8:3.

9.1 maintaining a list of application data associated with an application program;	Pearce, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Pearce, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Pearce, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Pearce discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> "One of the purposes of the operating system is to allocate resources to tasks (or "application programs," such as Word processors, spreadsheets, communications programs, database managers, games and the like and their associated data) as they are executed on the portable PC 100."

Pearce, 5:55-61.

> "The user again loads an application task and begins to edit a document. For purposes of this example, it is again assumed that the user leaves the portable PC 100 Without saving the document. A period of inactivity thus begins."

Pearce, 7:65-8:3.

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Pearce, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Pearce discloses this limitation:

> "During compression of the main memory, the record containing the identity of the allocable units compressed is also stored for use by the main memory restoration method detailed in FIG. 5."

Pearce, 8:27-30, Fig. 5.

> "As a part of the block 515, the units previously allocated to the reducing task are marked as unallocated and therefore free for subsequent allocation by the operating system."

Pearce, 8:67-9:3, Fig. 5.

See also Pearce 4:16-22

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Pearce, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Pearce discloses this limitation:

> "The most effective solution to power saving following a long period of user inactivity is to turn off the PC completely. However, it is unacceptable simply to interrupt has not yet been stored in non-volatile memory, particularly the hard disk drive. Rather, it is advantageous to store the contents of the volatile main memory of the PC as an image in the nonvolatile secondary storage device (a so-called "suspend-to-disk" or "STD"). When the user restores power to the PC, the image is restored to the main memory (a "resume-from-disk" or "RFD"), placing the PC in exactly the same functional state as it Was When power was interrupted."

Pearce, 1:65-2:10.

> "In a preferred embodiment, the suspend circuit further comprises a circuit for restoring the main memory by decompressing the compressed image into the main memory. The allocable units allocated to the reducing task are designated as unallocated units in the decompressed image."

Pearce, 3:47-52.

> "The present invention is designed to conserve computer power by storing the pertinent contents of the main memory 210 as a compressed image in the nonvolatile secondary storage device 260, allowing power to be interrupted to the portable PC 100 in its entirety. When the user so directs, power is restored to the PC, and the image is decompressed back into the main memory 210, allowing the CPU 200 to continue processing at the point where it stopped prior to interruption of power."

Pearce, 5:36-44.

> "If a compressed image file is detected, execution proceeds to a block 515, wherein a corresponding (e.g., run length decoding) decompression

> routine decompresses and restores the main memory 210 to the state in which it was prior to shutdown. In the first embodiment, the units previously allocated to the reducing task remain filled with the bit pattern. In the second embodiment, the stored record is retrieved and used as a guide by the decompression routine to restore the compressed allocable units to their proper position in main memory."

Pearce, 8:52-61, Fig. 5.

See also Pearce 4:16-22.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Pearce, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Pearce, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Pearce, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Pearce discloses this limitation:

See Claims 1.3 and 1.4 above.

See also

> "There are a wide variety of conventional data compression algorithms that are suitable to compress the contents of main memory to create a compressed image to be stored in the secondary storage device. Those skilled in the art are familiar With standard compression algorithms such as run length encoding, adaptive pattern substitution, variable length character encoding (such as Huffman coding), restricted variability codes, dictionary substitution, differencing and ordered data schemes."

Pearce, 7:11-19.

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Pearce, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Pearce discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

See also ***Look for plurality of encoders***

> "(3) a circuit for executing the data compression process to store a compressed image of the main memory in the secondary storage unit, the bit pattern allowing a size of the compressed image to be reduced and a time required to compress and store the compressed image to be minimized."

Pearce, 2:49-53. *See also* Pearce, 3:36-42, 9:25-31.

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Pearce, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Pearce, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Pearce, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Pearce, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Pearce, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Pearce, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claim 1.4 above.	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Pearce, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 4, 10 and 11 above.	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Pearce, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 15 and 22 above.	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Pearce, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 16 and 22 above.	

27.1 a boot device..	Pearce, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claim 1 above.	

27.2 a processor..	Pearce, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claim 1.2 above.	

27.3 cache memory; and.	Pearce, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 1.3 and 1.4 above.	

27.4 non-volatile memory for storing logic code for use by the processor,..	Pearce, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claim 1.1 and 1.3 above.	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Pearce, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claim 1.1 above.	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Pearce, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claim 1.3 above.	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Pearce, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claim 1.4 above.	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Pearce, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 4, 10, and 11 above.	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Pearce, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 15 and 27 above.	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Pearce, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Pearce discloses this limitation: *See* Claims 16 and 27 above.	

Appendix A16
Invalidity of U.S. Patent 7,181,608 based on Rahman

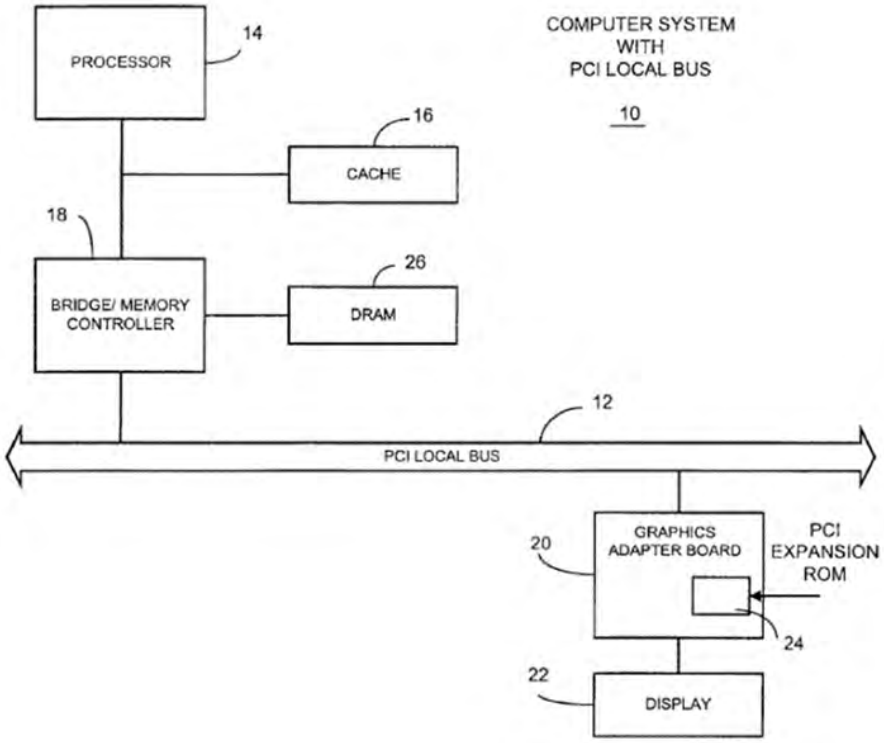
U.S. Patent No. 5,901,310 Rahman ("Rahman") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Rahman, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Rahman discloses this limitation:



Rahman, Fig. 1.

> "Initializing and configuring computer hardware with firmware stored in compressed form in nonvolatile semiconductor memory. Upon startup of the computer hardware, decompression software decompress the firmware, which is then stored in another memory. The computer hardware may be an adapter board (e.g., a graphics board connected to PCI bus), the nonvolatile semiconductor memory may be physically located on the adapter board, and the firmware may be firmware for

initializing and configuring the adapter board."

Rahman, Abstract.

> "The firmware may be the BIOS for initializing and configuring a personal computer."

Rahman, 2:1-2.

> "Devices that connect to the PCI bus provide initialization code and runtime code for the device. This code resides in ROM 24 on the device. When the computer system starts up or initializes, it detects the device, reads the code from ROM into the host system memory, such as a RAM or dynamic random access memory 26 (DRAM), and interprets the code."

Rahman, 2:51-57.

> "When the computer system starts up or initializes, it detects the adapter board connected to the PCI bus. It then locates the decompression code 29 in the adapter board's ROM. The computer system processor runs an FCode interpreter, which interprets the instructions in the decompression program as it reads the code from the adapter board's ROM. The compressed code 28 is decompressed, using the computer system's memory (such as RAM or DRAM 26) to store the image. After the image is decompressed, the system recognizes the decompressed image as a device driver that configures and initializes the adapter board, and supplies the runtime set of instructions for the adapter board."

Rahman, 2:66-3:11.

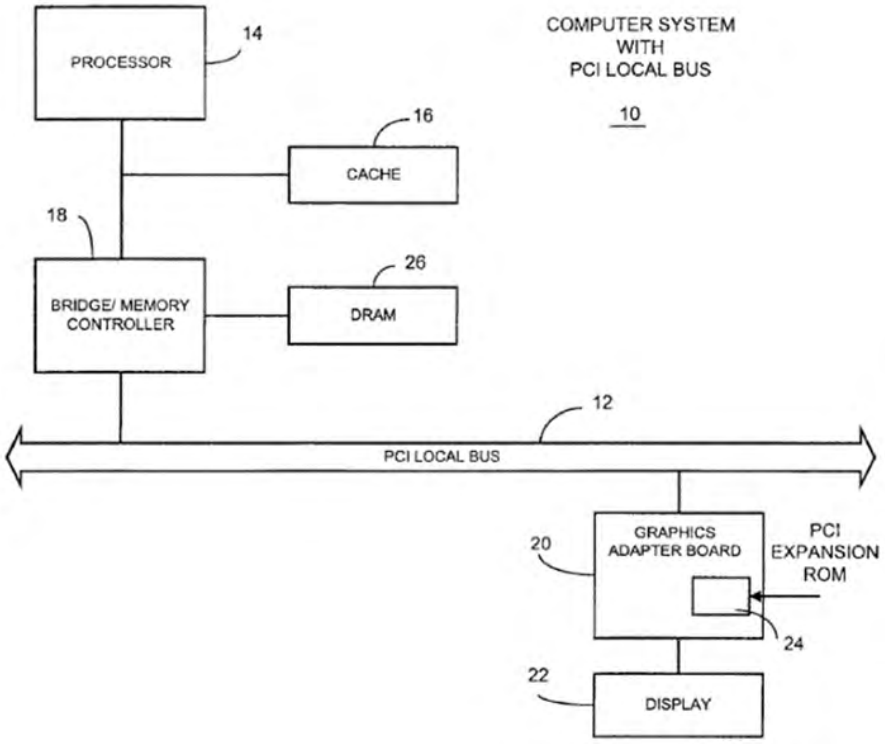
> "Other embodiments are within the scope of the following claims. For example, a personal computer's basic input/output system (BIOS) is firmware stored in ROM and could be stored in a compressed format. Furthermore, other types of nonvolatile semiconductor memory, such as programmable ROMs (PROMS) and erasable programmable ROMs (EPROMs), could be used to store the compressed firmware."

Rahman, 4:63-5:3.

1.1 maintaining a list of boot data used for booting a computer system;	Rahman, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Rahman discloses this limitation:



Rahman, Fig. 1.

> "Initializing and configuring computer hardware with firmware stored in compressed form in nonvolatile semiconductor memory. Upon startup of the computer hardware, decompression software decompress the firmware, which is then stored in another memory. The computer hardware may be an adapter board (e.g., a graphics board connected to PCI bus), the nonvolatile semiconductor memory may be physically located on the adapter board, and the firmware may be firmware for

initializing and configuring the adapter board."

Rahman, Abstract.

> "The firmware may be the BIOS for initializing and configuring a personal computer."

Rahman, 2:1-2.

> "Devices that connect to the PCI bus provide initialization code and runtime code for the device. This code resides in ROM 24 on the device. When the computer system starts up or initializes, it detects the device, reads the code from ROM into the host system memory, such as a RAM or dynamic random access memory 26 (DRAM), and interprets the code."

Rahman, 2:51-57.

> "When the computer system starts up or initializes, it detects the adapter board connected to the PCI bus. It then locates the decompression code 29 in the adapter board's ROM. The computer system processor runs an FCode interpreter, which interprets the instructions in the decompression program as it reads the code from the adapter board's ROM. The compressed code 28 is decompressed, using the computer system's memory (such as RAM or DRAM 26) to store the image. After the image is decompressed, the system recognizes the decompressed image as a device driver that configures and initializes the adapter board, and supplies the runtime set of instructions for the adapter board."

Rahman, 2:66-3:11.

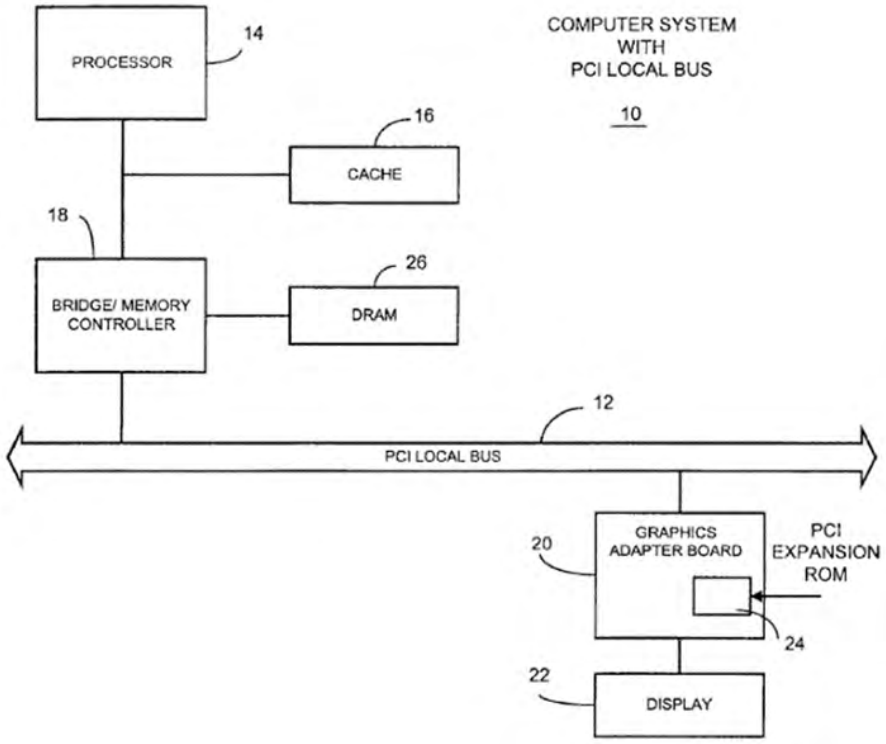
> "Other embodiments are within the scope of the following claims. For example, a personal computer's basic input/output system (BIOS) is firmware stored in ROM and could be stored in a compressed format. Furthermore, other types of nonvolatile semiconductor memory, such as programmable ROMs (PROMS) and erasable programmable ROMs (EPROMs), could be used to store the compressed firmware."

Rahman, 4:63-5:3.

1.2 initializing a central processing unit of the computer system;	Rahman, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Rahman discloses this limitation:



Rahman, Fig. 1.

> "Initializing and configuring computer hardware with firmware stored in compressed form in nonvolatile semiconductor memory. Upon startup of the computer hardware, decompression software decompress the firmware, which is then stored in another memory. The computer hardware may be an adapter board (e.g., a graphics board connected to PCI bus), the nonvolatile semiconductor memory may be physically located on the adapter board, and the firmware may be firmware for

initializing and configuring the adapter board."

Rahman, Abstract.

> "The firmware may be the BIOS for initializing and configuring a personal computer."

Rahman, 2:1-2.

> "Devices that connect to the PCI bus provide initialization code and runtime code for the device. This code resides in ROM 24 on the device. When the computer system starts up or initializes, it detects the device, reads the code from ROM into the host system memory, such as a RAM or dynamic random access memory 26 (DRAM), and interprets the code."

Rahman, 2:51-57.

> "When the computer system starts up or initializes, it detects the adapter board connected to the PCI bus. It then locates the decompression code 29 in the adapter board's ROM. The computer system processor runs an FCode interpreter, which interprets the instructions in the decompression program as it reads the code from the adapter board's ROM. The compressed code 28 is decompressed, using the computer system's memory (such as RAM or DRAM 26) to store the image. After the image is decompressed, the system recognizes the decompressed image as a device driver that configures and initializes the adapter board, and supplies the runtime set of instructions for the adapter board."

Rahman, 2:66-3:11.

> "Other embodiments are within the scope of the following claims. For example, a personal computer's basic input/output system (BIOS) is firmware stored in ROM and could be stored in a compressed format. Furthermore, other types of nonvolatile semiconductor memory, such as programmable ROMs (PROMS) and erasable programmable ROMs (EPROMs), could be used to store the compressed firmware."

Rahman, 4:63-5:3.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Rahman, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Rahman discloses this limitation:

> "Initializing and configuring computer hardware with firmware stored in compressed form in nonvolatile semiconductor memory."

Rahman, Abstract.

> "When the computer system starts up or initializes, it detects the adapter board connected to the PCI bus. It then locates the decompression code 29 in the adapter board's ROM. The computer system processor runs an FCode interpreter, which interprets the instructions in the decompression program as it reads the code from the adapter board's ROM. The compressed code 28 is decompressed, using the computer system's memory (such as RAM or DRAM 26) to store the image. After the image is decompressed, the system recognizes the decompressed image as a device driver that configures and initializes the adapter board, and supplies the runtime set of instructions for the adapter board."

Rahman, 2:66-3:11.

> "Other embodiments are within the scope of the following claims. For example, a personal computer's basic input/output system (BIOS) is firmware stored in ROM and could be stored in a compressed format. Furthermore, other types of nonvolatile semiconductor memory, such as programmable ROMs (PROMS) and erasable programmable ROMs (EPROMs), could be used to store the compressed firmware."

Rahman, 4:63-5:3.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Rahman, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Rahman discloses this limitation:

> "Upon startup of the computer hardware, decompression software decompress the firmware, which is then stored in another memory."

Rahman, Abstract.

> "The invention virtually increases the size of the nonvolatile semiconductor memory (e.g., a ROM) available for storing firmware by storing the firmware in compressed form and quickly decompressing it on startup. This permits the firmware memory to hold more instructions than would otherwise be possible. The invention is able to decompress firmware quickly, reliably, and fully automatically, so that the fact of the firmware being compressed is substantially invisible to the end-user."

Rahman, 1:48-56.

> "When the computer system starts up or initializes, it detects the adapter board connected to the PCI bus. It then locates the decompression code 29 in the adapter board's ROM. The computer system processor runs an

> FCode interpreter, which interprets the instructions in the decompression program as it reads the code from the adapter board's ROM. The compressed code 28 is decompressed, using the computer system's memory (such as RAM or DRAM 26) to store the image. After the image is decompressed, the system recognizes the decompressed image as a device driver that configures and initializes the adapter board, and supplies the runtime set of instructions for the adapter board."

Rahman, 2:66-3:11.

> "Other embodiments are within the scope of the following claims. For example, a personal computer's basic input/output system (BIOS) is firmware stored in ROM and could be stored in a compressed format. Furthermore, other types of nonvolatile semiconductor memory, such as programmable ROMs (PROMS) and erasable programmable ROMs (EPROMs), could be used to store the compressed firmware."

Rahman, 4:63-5:3.

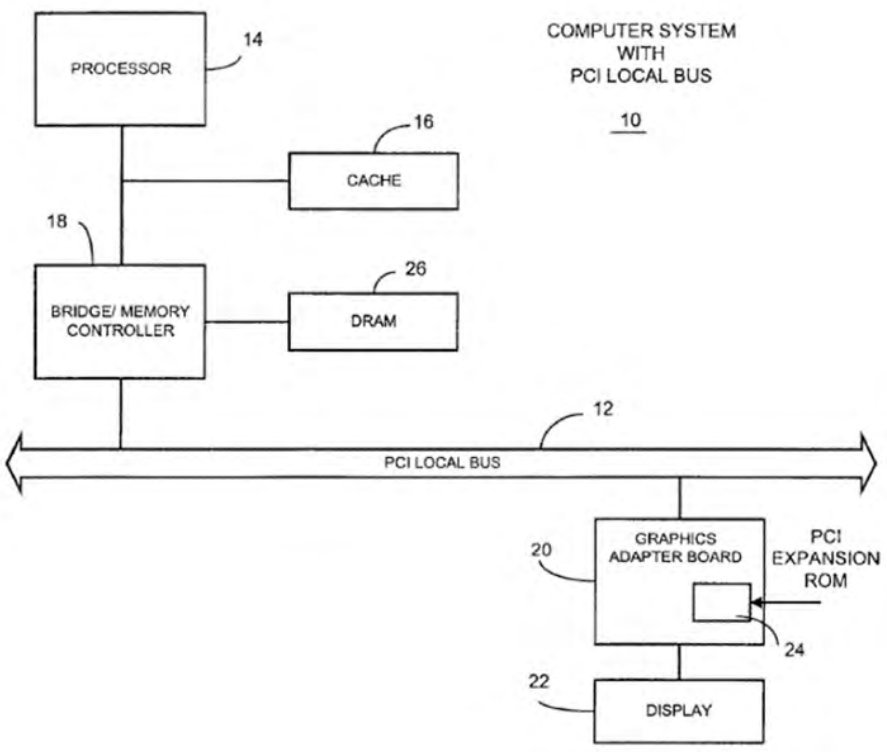
2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Rahman, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Rahman, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Rahman discloses this limitation:

See Claims 1.3, and 1.4 above.



Rahman, Fig. 1.

> "The processor accesses devices connected to the PCI local bus 12 through a bridge/memory controller 18."

Rahman, 2:47-48.

4. The method of claim 1, further comprising updating the list of boot data.	Rahman, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claim 1.1 above.	

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Rahman, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Rahman, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1.1 and 4 above.	

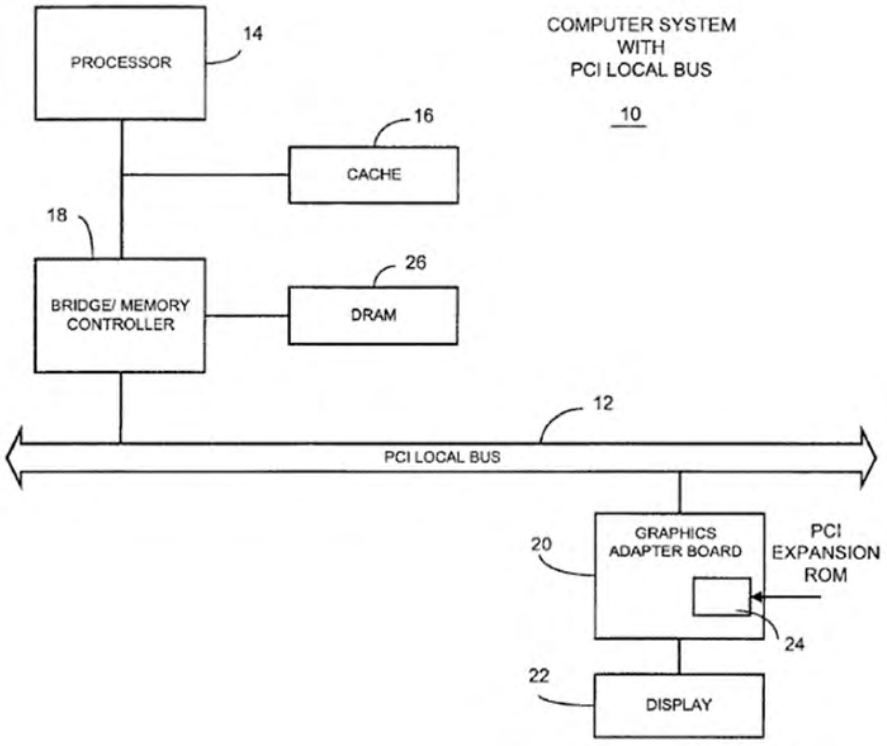
7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Rahman, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Rahman, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Rahman discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.



Rahman, Fig. 1.

> "The processor accesses devices connected to the PCI local bus 12 through a bridge/memory controller 18."

Rahman, 2:47-48.

7.2 a cache memory device; and;	Rahman, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Rahman, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Rahman, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Rahman, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Rahman, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.	

9.1 maintaining a list of application data associated with an application program;	Rahman, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

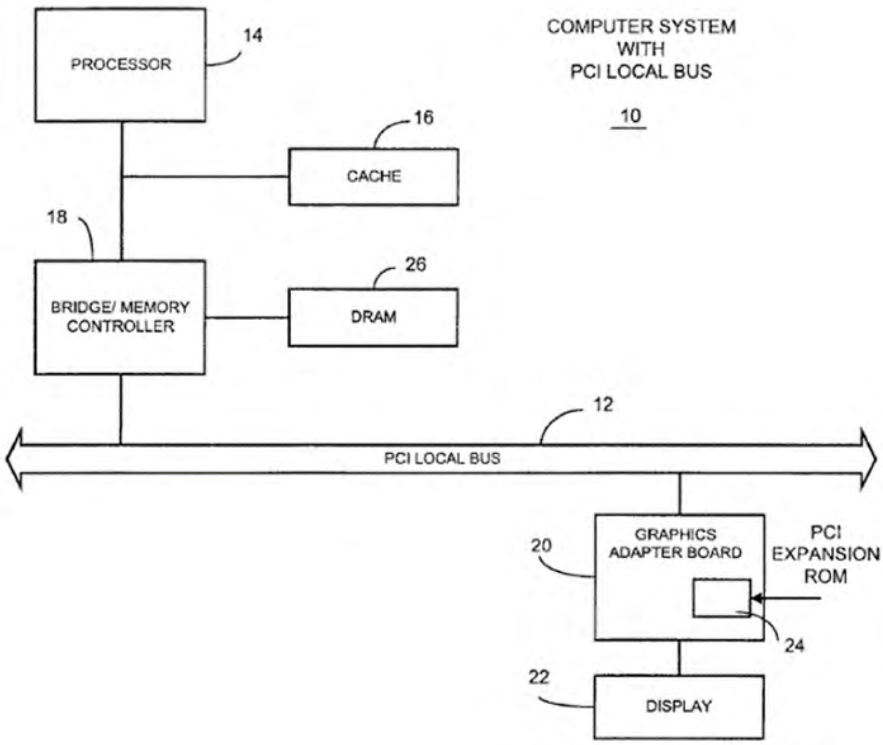
9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Rahman, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Rahman, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1.4, 2, and 8 above.	

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Rahman, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Rahman discloses this limitation:



Rahman, Fig. 1.

> "Initializing and configuring computer hardware with firmware stored in compressed form in nonvolatile semiconductor memory. Upon startup of the computer hardware, decompression software decompress the

> firmware, which is then stored in another memory. The computer hardware may be an adapter board (e.g., a graphics board connected to PCI bus), the nonvolatile semiconductor memory may be physically located on the adapter board, and the firmware may be firmware for initializing and configuring the adapter board."

Rahman, Abstract.

> "The firmware may be the BIOS for initializing and configuring a personal computer."

Rahman, 2:1-2.

> "Devices that connect to the PCI bus provide initialization code and runtime code for the device. This code resides in ROM 24 on the device. When the computer system starts up or initializes, it detects the device, reads the code from ROM into the host system memory, such as a RAM or dynamic random access memory 26 (DRAM), and interprets the code."

Rahman, 2:51-57.

> "When the computer system starts up or initializes, it detects the adapter board connected to the PCI bus. It then locates the decompression code 29 in the adapter board's ROM. The computer system processor runs an FCode interpreter, which interprets the instructions in the decompression program as it reads the code from the adapter board's ROM. The compressed code 28 is decompressed, using the computer system's memory (such as RAM or DRAM 26) to store the image. After the image is decompressed, the system recognizes the decompressed image as a device driver that configures and initializes the adapter board, and supplies the runtime set of instructions for the adapter board."

Rahman, 2:66-3:11.

> "Other embodiments are within the scope of the following claims. For example, a personal computer's basic input/output system (BIOS) is firmware stored in ROM and could be stored in a compressed format. Furthermore, other types of nonvolatile semiconductor memory, such as programmable ROMs (PROMS) and erasable programmable ROMs (EPROMs), could be used to store the compressed firmware."

Rahman, 4:63-5:3.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Rahman, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Rahman discloses this limitation:

> "Initializing and configuring computer hardware with firmware stored in compressed form in nonvolatile semiconductor memory."

Rahman, Abstract.

> "Upon startup of the computer hardware, decompression software decompress the firmware, which is then stored in another memory."

Rahman, Abstract.

> "The invention virtually increases the size of the nonvolatile semiconductor memory (e.g., a ROM) available for storing firmware by storing the firmware in compressed form and quickly decompressing it on startup. This permits the firmware memory to hold more instructions than would otherwise be possible. The invention is able to decompress firmware quickly, reliably, and fully automatically, so that the fact of the firmware being compressed is substantially invisible to the end-user."

Rahman, 1:48-56.

> "The firmware may be the BIOS for initializing and configuring a personal computer."

Rahman, 2:1-2.

> "When the computer system starts up or initializes, it detects the adapter board connected to the PCI bus. It then locates the decompression code 29 in the adapter board's ROM. The computer system processor runs an FCode interpreter, which interprets the instructions in the decompression program as it reads the code from the adapter board's ROM. The compressed code 28 is decompressed, using the computer system's

> memory (such as RAM or DRAM 26) to store the image. After the image is decompressed, the system recognizes the decompressed image as a device driver that configures and initializes the adapter board, and supplies the runtime set of instructions for the adapter board."

Rahman, 2:66-3:11.

> "Other embodiments are within the scope of the following claims. For example, a personal computer's basic input/output system (BIOS) is firmware stored in ROM and could be stored in a compressed format. Furthermore, other types of nonvolatile semiconductor memory, such as programmable ROMs (PROMS) and erasable programmable ROMs (EPROMs), could be used to store the compressed firmware."

Rahman, 4:63-5:3.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Rahman, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Rahman, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Rahman, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Rahman discloses this limitation:

See Claims 1.3 and 1.4 above.

> "The compression technique used may include both run-length encoding and pattern compression, and may operate at the bit level."

Rahman, Abstract.

> "Compression techniques using both run-length encoding and pattern compression may be used (e.g., the Ross Compression technique). The compression technique may scan an input file. First, it may determine if it can perform run-length-encoding compression on the current character. If not, the compression technique may determine if it can compress a pattern of characters. If the run-length-encoding and pattern matching Were not successful, the compression technique may copy the character directly to the compressed file."

Rahman, 2:7-16.

> "The compression and decompression techniques utilize four 3-byte formats. FIG. 3 illustrates the formats, which are a short run-length-encoded format 30, a long run-length encoded format 32, a short patterned format 34, and a long patterned format 36."

Rahman, 3:12-16. *See also* 3:17-17, Appendix.

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Rahman, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Rahman discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

> "The compression technique used may include both run-length encoding and pattern compression, and may operate at the bit level."

Rahman, Abstract.

> "Compression techniques using both run-length encoding and pattern compression may be used (e.g., the Ross Compression technique). The compression technique may scan an input file. First, it may determine if it can perform run-length-encoding compression on the current character. If not, the compression technique may determine if it can compress a pattern of characters. If the run-length-encoding and pattern matching Were not successful, the compression technique may copy the character directly to the compressed file."

Rahman, 2:7-16.

> "The compression and decompression techniques utilize four 3-byte formats. FIG. 3 illustrates the formats, which are a short run-length-encoded format 30, a long run-length encoded format 32, a short patterned format 34, and a long patterned format 36."

Rahman, 3:12-16. *See also* 3:17-17, Appendix.

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Rahman, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Rahman, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Rahman, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claim 1.1 above.	

22.2 initializing a central processing unit of the computer system;.	Rahman, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claim 1.2 above.	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Rahman, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claim 1.3 above.	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Rahman, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claim 1.4 above.	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Rahman, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 4, 10 and 11 above.	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Rahman, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 15 and 22 above.	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Rahman, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 16 and 22 above.	

27.1 a boot device..	Rahman, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claim 1 above.	

27.2 a processor..	Rahman, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claim 1.2 above.	

27.3 cache memory; and.	Rahman, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 1.3 and 1.4 above.	

27.4 non-volatile memory for storing logic code for use by the processor,..	Rahman, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claim 1.1 and 1.3 above.	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Rahman, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claim 1.1 above.	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Rahman, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claim 1.3 above.	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Rahman, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claim 1.4 above.	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Rahman, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 4, 10, and 11 above.	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Rahman, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 15 and 27 above.	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Rahman, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rahman discloses this limitation: *See* Claims 16 and 27 above.	

Appendix A17
Invalidity of U.S. Patent 7,181,608 based on Settsu

U.S. Patent No. 6,374,353 to Settsu ("Settsu") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

In addition, Apple incorporates by reference, as if set forth fully herein, all arguments related to Settsu in pending *inter partes* review petitions IPR2016-01737, IPR2016-01738, and IPR2016-01739.

Appendix A17
Invalidity of U.S. Patent 7,181,608 based on Settsu

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Settsu, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Settsu discloses this claim limitation:

> A method of booting up an information processing apparatus is provided. An operating system is divided into a mini operating system (OS) module having a function of bootstrap and an OS main body module having functions other than the function of bootstrap. The mini OS module can be located in a boot block of a boot device, whereas the OS main body module can be located in a file system of the boot device. A firmware or F/W code module stored in a ROM loads the mini OS module into memory when booting up the information processing apparatus. The mini OS module then loads the OS main body module into memory and then initializes the OS main body module.

Settsu, Abstract

> The present invention relates to an information processing apparatus capable of reducing the time required for booting itself when it is powered on, and a method of booting an information processing apparatus at a high speed.

Settsu, 1:8-12.

See also Settsu, 1:43-2:25, 3:6-25, 10:43-12:16, 13:49-15:4.

1.1 maintaining a list of boot data used for booting a computer system;	Settsu, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

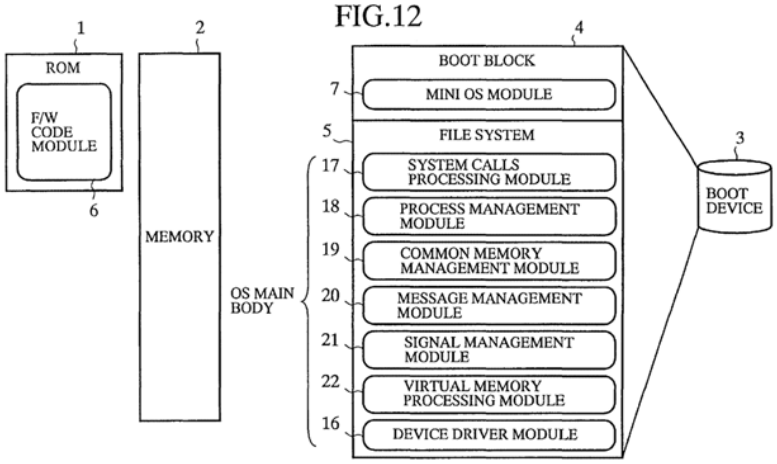
Settsu discloses this claim limitation:

> Referring next to FIG. 1, there is illustrated a block diagram showing the structure of an information processing apparatus according to a first embodiment of th present invention. In the figure, reference numeral 1 denotes a ROM of the information processing apparatus, 2 denotes a memory of the information processing apparatus, 3 denotes 5 a boot device of the information processing apparatus, 4 denotes a boot block in the boot device 3, 5 denotes a file system I the boot device 3, and 6 denotes a firmware or FI/W code module stored in the ROM 1. The FI/W code module 6 is directly executed on the ROM 1 so as to load data from the boot block 4 in the boot device 3 into the memory 2 and then assume that the loaded data is a code and execute the code after setting up and running diagnostic checks on a hardware or H/W register. Further, reference numeral 7 denotes a mini operating system (OS) module complied and liked in the same way as ordinary program files and located in the boot block 4 within the boot device, the mini OS module having OS functions required for bootstrap processing, and 8 denotes an OS main body module located in the file system 5 within the boot device and provided with OS functions except the OS functions included in the mini OS module 7. When the information processing apparatus is powered on, it goes through initialization and transfers control to the F/W code module 6 stored in the ROM 1.
>
> Referring next to FIG. 2, there is illustrated a diagram showing the structure of the mini OS module 7 stored in the boot block of the boot device of the information processing apparatus according to the first embodiment of the present invention. As shown in the figure, the mini OS module 7 consists of a mini kernel module 9 which is a basic part of the OS, a boot device driver module 10 for performing I/O operations on the boot device 3, and an OS loading and initialization processing module 11 for loading the OS main body module 8 from the boot device

3 into the memory 2 and for executing the initialization of the OS main body module 8.

Settsu, 7:65-8:35

FIG.12



Settsu, Fig. 12

Referring next to FIG. 12, there is illustrated a block diagram showing the structure of an information processing apparatus according to a fourth embodiment of the present invention. In the figure, the same reference numerals as shown in FIG. 5 designate the same or like elements, and therefore the description of those elements will be omitted hereinafter. Like the OS of the second embodiment, the OS of the second embodiment is divided into a mini OS module 7 and a main body of the OS, and the main body is further divided into a plurality of functional modules, such as a system call processing module 17, a process management module 18, a common memory management module 19, a message management module 20, a signal management module 21, a virtual memory processing module 22, and a device driver module 16. The plurality of functional modules are separately stored as compressed files in a file system 5 of a boot device 3.

Settsu, 13:55-65

See also Settsu, 16:7-17:62, and Figs. 1-4, 6-9, 13-14, 20 & 35-36.

1.2 initializing a central processing unit of the computer system;	Settsu, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Settsu discloses this claim limitation:

> A method of booting up an information processing apparatus is provided. An operating system is divided into a mini operating system (OS) module having a function of bootstrap and an OS main body module having functions other than the function of bootstrap. The mini OS module can be located in a boot block of a boot device, whereas the OS main body module can be located in a file system of the boot device. A firmware or F/W code module stored in a ROM loads the mini OS module into memory when booting up the information processing apparatus. The mini OS module then loads the OS main body module into memory and then initializes the OS main body module.

Settsu, Abstract

> The present invention relates to an information processing apparatus capable of reducing the time required for booting itself when it is powered on, and a method of booting an information processing apparatus at a high speed.

Settsu, 1:8-12.

> Referring next to FIG. 1, there is illustrated a block diagram showing the structure of an information processing apparatus according to a first embodiment of th present invention. In the figure, reference numeral 1 denotes a ROM of the information processing apparatus, 2 denotes a memory of the information processing apparatus, 3 denotes 5 a boot device of the information processing apparatus, 4 denotes a boot block in the boot device 3, 5 denotes a file system I the boot device 3, and 6 denotes a firmware or FI/W code module stored in the ROM 1. The FI/W code module 6 is directly executed on the ROM 1 so as to load data from the boot block 4 in the boot device 3 into the memory 2 and

then assume that the loaded data is a code and execute the code after setting up and running diagnostic checks on a hardware or H/W register. Further, reference numeral 7 denotes a mini operating system (OS) module complied and liked in the same way as ordinary program files and located in the boot block 4 within the boot device, the mini OS module having OS functions required for bootstrap processing, and 8 denotes an OS main body module located in the file system 5 within the boot device and provided with OS functions except the OS functions included in the mini OS module 7. When the information processing apparatus is powered on, it goes through initialization and transfers control to the F/W code module 6 stored in the ROM 1.

Referring next to FIG. 2, there is illustrated a diagram showing the structure of the mini OS module 7 stored in the boot block of the boot device of the information processing apparatus according to the first embodiment of the present invention. As shown in the figure, the mini OS module 7 consists of a mini kernel module 9 which is a basic part of the OS, a boot device driver module 10 for performing I/O operations on the boot device 3, and an OS loading and initialization processing module 11 for loading the OS main body module 8 from the boot device 3 into the memory 2 and for executing the initialization of the OS main body module 8.

Settsu, 7:65-8:35

Referring next to FIG. 3, there is illustrated a diagram showing the structure of the OS main body module 8 of the information processing apparatus according to the first embodiment of the present invention. As shown in the figure, the OS main body module 8 consists of a kernel module 15 having kernel functions except the functions included in the mini kernel module 9 of FIG. 2, and a device driver module 16 for performing I/O operations on devices (not shown) except the boot device 3. . . . The OS main body module 8 is located within the file system 5 of the boot device 3 and is loaded into the memory 2 by the mini OS module 7. The OS main body module 8 then goes through initialization.

Settsu, 8:47-9:3

Referring next to FIG. 4, there is illustrated a flow chart showing operations of the mini OS module 7 of the information processing apparatus according to the first embodiment of the present invention. The F/W code module 6 loads the mini OS module 7 into the memory 2 and transfers control to the mini OS module 7. The mini OS module 7 then, in step ST101, executes initialization of the mini kernel module 9.

> In this case, the thread management module 12, the I/O management module 13, and the thread communication management module 14 are initialized and their functions are available now. Next, the mini OS module 7, in step ST102, executes initialization of the boot device driver module 10. The boot device driver module 10 registers an interrupt request from the boot device into an interrupt table of the I/O management module 13 so as to handle the interrupt request, and generates and starts execution of a thread for boot device I/O processing by using the thread management module 12.

Settsu, 9:7-21.

> The mini OS module 7, in step ST103, generates a thread for the OS loading and initialization processing module 11 by using the thread management module 12. The mini OS module 7 further, in step ST104, starts execution of the thread by using the thread management module 12. The OS loading and initialization processing module 11 is thus started up as the thread. After that, the mini OS module 7 transfers control to the OS loading and initialization processing module 11.

Settsu, 9:30-39.

> As previously mentioned, in accordance with the second embodiment of the present invention, the main body of the OS is divided into a plurality of functional modules according a plurality of functions to be performed by the main body of the OS. Further, the plurality of functional modules are separately stored in the file system 5. In addition, the OS load processing and the OS initialization processing can be performed in parallel with each other after any one of the plurality of functional modules of the OS main body is loaded into the memory. As a result, while the CPU waits for the occurrence of an event in performing the OS load or initialization processing, the CPU does not idle but the CPU performs another processing. Accordingly, the second embodiment of the present invention provides an advantage of being able to further reduce the time required for booting up the information processing apparatus.

Settsu, 12:1-16.

> Referring next to FIG. 13, there is illustrated a block diagram showing the structure of the mini OS module 7 of the information processing apparatus according to the fourth embodiment of the present invention. Like the mini OS module 7 of the second embodiment as shown in FIG. 6, the mini OS module 7 of the fourth embodiment is provided with a mini kernel module 9, a boot device driver module 10, and an OS

initialization processing module 31. The mini OS module 7 of the fourth embodiment further comprises an OS loading and decompression processing module 50 having a function of decompressing a loaded functional module in addition to the function of the OS load processing module 30 of the second embodiment, instead of the OS load processing module 30.

Settsu, 13:66-14:12

See also Figs. 1-4, 6-9, 12-14, 20 & 35-36.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Settsu, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Settsu discloses this claim limitation:

> A method of booting up an information processing apparatus is provided. An operating system is divided into a mini operating system (OS) module having a function of bootstrap and an OS main body module having functions other than the function of bootstrap. The mini OS module can be located in a boot block of a boot device, whereas the OS main body module can be located in a file system of the boot device. A firmware or F/W code module stored in a ROM loads the mini OS module into memory when booting up the information processing apparatus. The mini OS module then loads the OS main body module into memory and then initializes the OS main body module.

Settsu, Abstract

> The present invention relates to an information processing apparatus capable of reducing the time required for booting itself when it is powered on, and a method of booting an information processing apparatus at a high speed.

Settsu, 1:8-12.

> In accordance with an aspect of the present invention, there is provided an information processing apparatus comprising: a boot device divided into a boot block in which a mini operating system (OS) module having a function required for bootstrap processing is located and a file system

> in which an operating system (OS) main body module having functions other than the function of bootstrap. . . .

Settsu, 1:51-57

> In accordance with another preferred embodiment of the present invention, the plurality of functional modules, into which the OS main body module is divided, are stored as compressed files in the file system and the loading and initialization processing module of the mini OS module is divided into an OS loading and decompression processing module and an OS initialization module. Further, the mini OS module generates and starts execution of a thread for the OS loading and decompression processing module after the mini OS module initializes the mini kernel module and the boot device driver module. After the thread for the OS loading and decompression processing module is started, the OS loading and decompression processing module loads each of the plurality of functional modules into the memory and decompresses the loaded functional module, and then generates and starts execution of a thread for the OS initialization module. After the thread for the OS initialization module is executed, the OS initialization module initializes each of the plurality of functional modules loaded into the memory and decompressed.

Settsu, 3:6-25

> Referring next to FIG. 1, there is illustrated a block diagram showing the structure of an information processing apparatus according to a first embodiment of th present invention. In the figure, reference numeral 1 denotes a ROM of the information processing apparatus, 2 denotes a memory of the information processing apparatus, 3 denotes 5 a boot device of the information processing apparatus, 4 denotes a boot block in the boot device 3, 5 denotes a file system I the boot device 3, and 6 denotes a firmware or FI/W code module stored in the ROM 1. The FI/W code module 6 is directly executed on the ROM 1 so as to load data from the boot block 4 in the boot device 3 into the memory 2 and then assume that the loaded data is a code and execute the code after setting up and running diagnostic checks on a hardware or H/W register. Further, reference numeral 7 denotes a mini operating system (OS) module complied and liked in the same way as ordinary program files and located in the boot block 4 within the boot device, the mini OS module having OS functions required for bootstrap processing, and 8 denotes an OS main body module located in the file system 5 within the boot device and provided with OS functions except the OS functions included in the mini OS module 7. When the information processing

apparatus is powered on, it goes through initialization and transfers control to the F/W code module 6 stored in the ROM 1.

Referring next to FIG. 2, there is illustrated a diagram showing the structure of the mini OS module 7 stored in the boot block of the boot device of the information processing apparatus according to the first embodiment of the present invention. As shown in the figure, the mini OS module 7 consists of a mini kernel module 9 which is a basic part of the OS, a boot device driver module 10 for performing I/O operations on the boot device 3, and an OS loading and initialization processing module 11 for loading the OS main body module 8 from the boot device 3 into the memory 2 and for executing the initialization of the OS main body module 8.

Settsu, 7:65-8:35

Referring next to FIG. 3, there is illustrated a diagram showing the structure of the OS main body module 8 of the information processing apparatus according to the first embodiment of the present invention. As shown in the figure, the OS main body module 8 consists of a kernel module 15 having kernel functions except the functions included in the mini kernel module 9 of FIG. 2, and a device driver module 16 for performing I/O operations on devices (not shown) except the boot device 3. . . . The OS main body module 8 is located within the file system 5 of the boot device 3 and is loaded into the memory 2 by the mini OS module 7. The OS main body module 8 then goes through initialization.

Settsu, 8:47-9:3

Referring next to FIG. 4, there is illustrated a flow chart showing operations of the mini OS module 7 of the information processing apparatus according to the first embodiment of the present invention. The F/W code module 6 loads the mini OS module 7 into the memory 2 and transfers control to the mini OS module 7. The mini OS module 7 then, in step ST101, executes initialization of the mini kernel module 9. In this case, the thread management module 12, the I/O management module 13, and the thread communication management module 14 are initialized and their functions are available now. Next, the mini OS module 7, in step ST102, executes initialization of the boot device driver module 10. The boot device driver module 10 registers an interrupt request from the boot device into an interrupt table of the I/O management module 13 so as to handle the interrupt request, and generates and starts execution of a thread for boot device I/O processing by using the thread management module 12.

Settsu, 9:7-21.

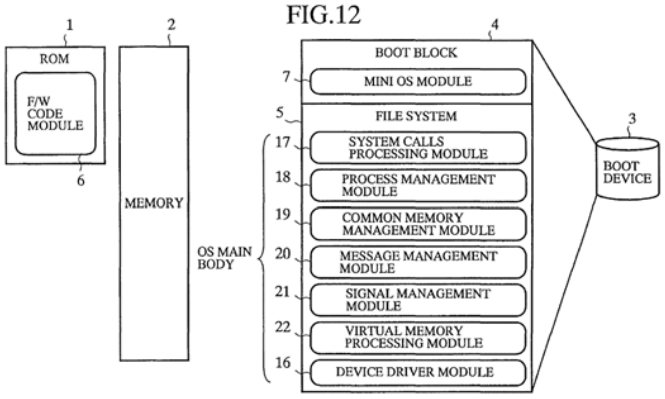
> The mini OS module 7, in step ST103, generates a thread for the OS loading and initialization processing module 11 by using the thread management module 12. The mini OS module 7 further, in step ST104, starts execution of the thread by using the thread management module 12. The OS loading and initialization processing module 11 is thus started up as the thread. After that, the mini OS module 7 transfers control to the OS loading and initialization processing module 11.

Settsu, 9:30-39.

> As previously mentioned, in accordance with the second embodiment of the present invention, the main body of the OS is divided into a plurality of functional modules according a plurality of functions to be performed by the main body of the OS. Further, the plurality of functional modules are separately stored in the file system 5. In addition, the OS load processing and the OS initialization processing can be performed in parallel with each other after any one of the plurality of functional modules of the OS main body is loaded into the memory. As a result, while the CPU waits for the occurrence of an event in performing the OS load or initialization processing, the CPU does not idle but the CPU performs another processing. Accordingly, the second embodiment of the present invention provides an advantage of being able to further reduce the time required for booting up the information processing apparatus.

Settsu, 12:1-16.

FIG.12



Settsu, Fig. 12

Referring next to FIG. 12, there is illustrated a block diagram showing the structure of an information processing apparatus according to a fourth embodiment of the present invention. In the figure, the same reference numerals as shown in FIG. 5 designate the same or like elements, and therefore the description of those elements will be omitted hereinafter. Like the OS of the second embodiment, the OS of the second embodiment is divided into a mini OS module 7 and a main body of the OS, and the main body is further divided into a plurality of functional modules, such as a system call processing module 17, a process management module 18, a common memory management module 19, a message management module 20, a signal management module 21, a virtual memory processing module 22, and a device driver module 16. The plurality of functional modules are separately stored as compressed files in a file system 5 of a boot device 3.

Settsu, 13:55-65

Referring next to FIG. 13, there is illustrated a block diagram showing the structure of the mini OS module 7 of the information processing apparatus according to the fourth embodiment of the present invention. Like the mini OS module 7 of the second embodiment as shown in FIG. 6, the mini OS module 7 of the fourth embodiment is provided with a mini kernel module 9, a boot device driver module 10, and an OS initialization processing module 31. The mini OS module 7 of the fourth embodiment further comprises an OS loading and decompression processing module 50 having a function of decompressing a loaded functional module in addition to the function of the OS load processing module 30 of the second embodiment, instead of the OS load processing module 30.

Settsu, 13:66-14:12

Referring next to FIG. 14, there is illustrated a flow chart showing operations of the OS loading and decompression processing module 50 of the information processing apparatus according to the fourth embodiment of the present invention. The OS loading and decompression processing module 50 to which control from the mini OS module 7 has been transferred, in step ST181, loads the main body of the OS stored in the file system 5 of the boot device 3, such as the system call processing module 17, the process management module 18, the common memory management module 19, the message management module 20, the signal management module 21, the virtual memory processing module 22, and the device driver module 16, into the memory 2. In performing step ST181, the OS loading and decompression processing module 50 loads any one of those functional

> modules 16 to 22 first. The OS loading and decompression processing module 50 then, in step ST182, decompresses one functional module loaded into the memory. Since the plurality of functional modules 16 to 22 are stored as compresses files in the file system 5 of the boot device, the functional module loaded into the memory 2 is compressed data. Therefore, in performing step ST182, the OS loading and decompression processing module 50 decompresses the compressed data so as to convert it into executable code and data.

Settsu, 14:13-37

> As previously mentioned, in accordance with the fourth embodiment of the present invention, the main body of the OS is divided into a plurality of functional modules according a plurality of functions to be performed by the main body, and the plurality of functional modules are stored as compressed files in the file system **5** of the boot device. Further, the OS loading and decompression processing module **50** decompresses each of the plurality of functional modules each time it loads each of them into memory. As a result, the time required for I/O processing can be reduced. Accordingly, the fourth embodiment of the present invention provides an advantage of being able to further reduce the time required for booting up the information processing apparatus.

Settsu, 14:58-15:5

See also Settsu, 8:21-35, 8:66-9:11, 11:7-9, 11:18-39, 13:49-15:5, 16:7-17:62, and Figs. 1-4, 6-9, 13-14, 20 & 35-36.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Settsu, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Settsu discloses this claim limitation:

> A method of booting up an information processing apparatus is provided. An operating system is divided into a mini operating system (OS) module having a function of bootstrap and an OS main body module having functions other than the function of bootstrap. The mini OS module can be located in a boot block of a boot device, whereas the OS main body module can be located in a file system of the boot device. A firmware or F/W code module stored in a ROM loads the mini OS module into memory when booting up the information processing apparatus. The mini OS module then loads the OS main body module into memory and then initializes the OS main body module.

Settsu, Abstract

> The present invention relates to an information processing apparatus capable of reducing the time required for booting itself when it is powered on, and a method of booting an information processing apparatus at a high speed.

Settsu, 1:8-12.

> In accordance with an aspect of the present invention, there is provided an information processing apparatus comprising: a boot device divided into a boot block in which a mini operating system (OS) module having a function required for bootstrap processing is located and a file system in which an operating system (OS) main body module having functions other than the function of bootstrap. . . .

Settsu, 1:51-57

> In accordance with another preferred embodiment of the present invention, the plurality of functional modules, into which the OS main body module is divided, are stored as compressed files in the file system and the loading and initialization processing module of the mini OS module is divided into an OS loading and decompression processing module and an OS initialization module. Further, the mini OS module generates and starts execution of a thread for the OS loading and decompression processing module after the mini OS module initializes the mini kernel module and the boot device driver module. After the thread for the OS loading and decompression processing module is started, the OS loading and decompression processing module loads each of the plurality of functional modules into the memory and decompresses the loaded functional module, and then generates and starts execution of a thread for the OS initialization module. After the thread for the OS initialization module is executed, the OS initialization module initializes each of the plurality of functional modules loaded into the memory and decompressed.

Settsu, 3:6-25

> Referring next to FIG. 1, there is illustrated a block diagram showing the structure of an information processing apparatus according to a first embodiment of th present invention. In the figure, reference numeral 1 denotes a ROM of the information processing apparatus, 2 denotes a memory of the information processing apparatus, 3 denotes 5 a boot device of the information processing apparatus, 4 denotes a boot block in the boot device 3, 5 denotes a file system I the boot device 3, and 6 denotes a firmware or FI/W code module stored in the ROM 1. The FI/W code module 6 is directly executed on the ROM 1 so as to load data from the boot block 4 in the boot device 3 into the memory 2 and then assume that the loaded data is a code and execute the code after setting up and running diagnostic checks on a hardware or H/W register. Further, reference numeral 7 denotes a mini operating system (OS)

module complied and liked in the same way as ordinary program files and located in the boot block 4 within the boot device, the mini OS module having OS functions required for bootstrap processing, and 8 denotes an OS main body module located in the file system 5 within the boot device and provided with OS functions except the OS functions included in the mini OS module 7. When the information processing apparatus is powered on, it goes through initialization and transfers control to the F/W code module 6 stored in the ROM 1.

Referring next to FIG. 2, there is illustrated a diagram showing the structure of the mini OS module 7 stored in the boot block of the boot device of the information processing apparatus according to the first embodiment of the present invention. As shown in the figure, the mini OS module 7 consists of a mini kernel module 9 which is a basic part of the OS, a boot device driver module 10 for performing I/O operations on the boot device 3, and an OS loading and initialization processing module 11 for loading the OS main body module 8 from the boot device 3 into the memory 2 and for executing the initialization of the OS main body module 8.

Settsu, 7:65-8:35

Referring next to FIG. 3, there is illustrated a diagram showing the structure of the OS main body module 8 of the information processing apparatus according to the first embodiment of the present invention. As shown in the figure, the OS main body module 8 consists of a kernel module 15 having kernel functions except the functions included in the mini kernel module 9 of FIG. 2, and a device driver module 16 for performing I/O operations on devices (not shown) except the boot device 3. . . . The OS main body module 8 is located within the file system 5 of the boot device 3 and is loaded into the memory 2 by the mini OS module 7. The OS main body module 8 then goes through initialization.

Settsu, 8:47-9:3

Referring next to FIG. 4, there is illustrated a flow chart showing operations of the mini OS module 7 of the information processing apparatus according to the first embodiment of the present invention. The F/W code module 6 loads the mini OS module 7 into the memory 2 and transfers control to the mini OS module 7. The mini OS module 7 then, in step ST101, executes initialization of the mini kernel module 9. In this case, the thread management module 12, the I/O management module 13, and the thread communication management module 14 are

> initialized and their functions are available now. Next, the mini OS module 7, in step ST102, executes initialization of the boot device driver module 10. The boot device driver module 10 registers an interrupt request from the boot device into an interrupt table of the I/O management module 13 so as to handle the interrupt request, and generates and starts execution of a thread for boot device I/O processing by using the thread management module 12.

Settsu, 9:7-21.

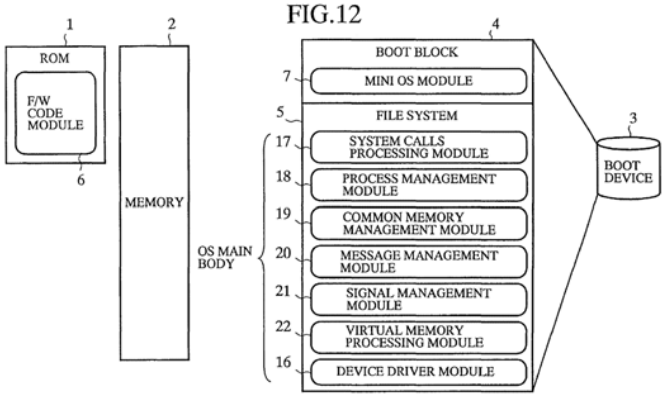
> The mini OS module 7, in step ST103, generates a thread for the OS loading and initialization processing module 11 by using the thread management module 12. The mini OS module 7 further, in step ST104, starts execution of the thread by using the thread management module 12. The OS loading and initialization processing module 11 is thus started up as the thread. After that, the mini OS module 7 transfers control to the OS loading and initialization processing module 11.

Settsu, 9:30-39.

> As previously mentioned, in accordance with the second embodiment of the present invention, the main body of the OS is divided into a plurality of functional modules according a plurality of functions to be performed by the main body of the OS. Further, the plurality of functional modules are separately stored in the file system 5. In addition, the OS load processing and the OS initialization processing can be performed in parallel with each other after any one of the plurality of functional modules of the OS main body is loaded into the memory. As a result, while the CPU waits for the occurrence of an event in performing the OS load or initialization processing, the CPU does not idle but the CPU performs another processing. Accordingly, the second embodiment of the present invention provides an advantage of being able to further reduce the time required for booting up the information processing apparatus.

Settsu, 12:1-16.

FIG.12



Settsu, Fig. 12

> Referring next to FIG. 12, there is illustrated a block diagram showing the structure of an information processing apparatus according to a fourth embodiment of the present invention. In the figure, the same reference numerals as shown in FIG. 5 designate the same or like elements, and therefore the description of those elements will be omitted hereinafter. Like the OS of the second embodiment, the OS of the second embodiment is divided into a mini OS module 7 and a main body of the OS, and the main body is further divided into a plurality of functional modules, such as a system call processing module 17, a process management module 18, a common memory management module 19, a message management module 20, a signal management module 21, a virtual memory processing module 22, and a device driver module 16. The plurality of functional modules are separately stored as compressed files in a file system 5 of a boot device 3.

Settsu, 13:55-65

> Referring next to FIG. 13, there is illustrated a block diagram showing the structure of the mini OS module 7 of the information processing apparatus according to the fourth embodiment of the present invention. Like the mini OS module 7 of the second embodiment as shown in FIG. 6, the mini OS module 7 of the fourth embodiment is provided with a mini kernel module 9, a boot device driver module 10, and an OS initialization processing module 31. The mini OS module 7 of the fourth embodiment further comprises an OS loading and decompression processing module 50 having a function of decompressing a loaded functional module in addition to the function of the OS load processing module 30 of the second embodiment, instead of the OS load processing module 30.

Settsu Claim 1.4

"servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

Settsu, 13:66-14:12

> Referring next to FIG. 14, there is illustrated a flow chart showing operations of the OS loading and decompression processing module 50 of the information processing apparatus according to the fourth embodiment of the present invention. The OS loading and decompression processing module 50 to which control from the mini OS module 7 has been transferred, in step ST181, loads the main body of the OS stored in the file system 5 of the boot device 3, such as the system call processing module 17, the process management module 18, the common memory management module 19, the message management module 20, the signal management module 21, the virtual memory processing module 22, and the device driver module 16, into the memory 2. In performing step ST181, the OS loading and decompression processing module 50 loads any one of those functional modules 16 to 22 first. The OS loading and decompression processing module 50 then, in step ST182, decompresses one functional module loaded into the memory. Since the plurality of functional modules 16 to 22 are stored as compresses files in the file system 5 of the boot device, the functional module loaded into the memory 2 is compressed data. Therefore, in performing step ST182, the OS loading and decompression processing module 50 decompresses the compressed data so as to convert it into executable code and data.

Settsu, 14:13-37

> As previously mentioned, in accordance with the fourth embodiment of the present invention, the main body of the OS is divided into a plurality of functional modules according a plurality of functions to be performed by the main body, and the plurality of functional modules are stored as compressed files in the file system **5** of the boot device. Further, the OS loading and decompression processing module **50** decompresses each of the plurality of functional modules each time it loads each of them into memory. As a result, the time required for I/O processing can be reduced. Accordingly, the fourth embodiment of the present invention provides an advantage of being able to further reduce the time required for booting up the information processing apparatus.

Settsu, 14:58-15:5

See also Settsu, 8:21-35, 8:66-9:11, 11:7-9, 11:18-39, 13:49-15:5, 16:7-17:62, and Figs. 1-4, 6-9, 13-14, 20 & 35-36.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Settsu, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Settsu discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> The present invention is made to overcome the above problems. It is therefore an object of the present invention to provide an information processing apparatus and a method capable of reducing the time required for booting up itself when it is powered on, and also reducing the time required to start execution of applications to be started automatically when the information processing apparatus is booted up.

Settsu, 1:44-50

> In accordance with another preferred embodiment of the present invention, the OS loading processing module of the mini OS module is an application (AP) execution and OS loading processing module for starting execution of at least a predetermined application module which is located in the file system and which can automatically be started and run on the operating system when booting up the information processing apparatus, and for loading each of the plurality of functional modules into the memory. Further, the predetermined application module includes a function definition file in which some functional modules required for the application module to run on the operating system are listed. After the mini OS module is loaded into the memory, the mini OS module initializes the mini kernel module and the boot device driver module and then generates and starts execution of a thread for the AP execution and OS loading processing module. After the thread for the

AP execution and OS loading processing module is started, the AP execution and OS loading processing module loads the application nodule from the file system into the memory and further loads some functional modules required for the application module into the memory according to the function definition file included in the application module, and then generates and starts execution of a thread for the OS initialization module. After the thread for the OS initialization module is started, the OS initialization module then initializes each of the some functional modules loaded into the memory. And, after the initialization of all of the some functional modules is completed, the application execution and OS loading processing module further loads the remainder of all functional modules included in the OS main body module into the memory and initializes the remainder using the OS initialization processing module while starting execution of the application module as a process.

Settsu, 3:48-4:14

Settsu, Fig. 18.

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Settsu, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 1.3, and 1.4 above.	

4. The method of claim 1, further comprising updating the list of boot data.	Settsu, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Settsu discloses this limitation:

See Claim 1.1 above.

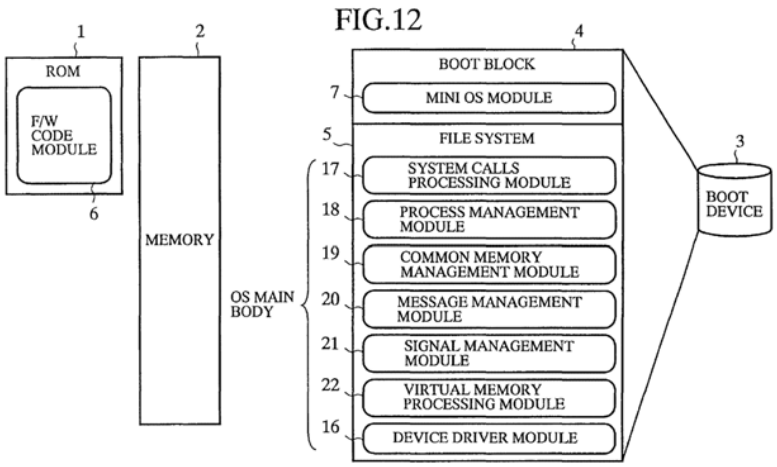
See also

> A method of booting up an information processing apparatus is provided. An operating system is divided into a mini operating system (OS) module having a function of bootstrap and an OS main body module having functions other than the function of bootstrap. The mini OS module can be located in a boot block of a boot device, whereas the OS main body module can be located in a file system of the boot device. A firmware or F/W code module stored in a ROM loads the mini OS module into memory when booting up the information processing apparatus. The mini OS module then loads the OS main body module into memory and then initializes the OS main body module.

Settsu, Abstract

> In accordance with another preferred embodiment of the present invention, the mini OS module further includes an address resolve table used for linking the mini OS module with the OS main body module. Further, after the mini OS module generates and starts execution of a thread for the OS loading and initialization processing module, the OS loading and initialization processing module loads the OS main body module into the memory and then initializes it, loads a first process to be executed first, into the memory, loads code portions of the mini kernel module and the boot device driver module into the memory, and writes addresses of the code portions loaded into the memory into the address resolve table.

Settsu, 5:39-51



Settsu, Fig. 12

> Referring next to FIG. 12, there is illustrated a block diagram showing the structure of an information processing apparatus according to a fourth embodiment of the present invention. In the figure, the same reference numerals as shown in FIG. 5 designate the same or like elements, and therefore the description of those elements will be omitted hereinafter. Like the OS of the second embodiment, the OS of the second embodiment is divided into a mini OS module 7 and a main body of the OS, and the main body is further divided into a plurality of functional modules, such as a system call processing module 17, a process management module 18, a common memory management module 19, a message management module 20, a signal management module 21, a virtual memory processing module 22, and a device driver module 16. The plurality of functional modules are separately stored as compressed files in a file system 5 of a boot device 3.

Settsu, 13:55-65

> The OS loading and decompression processing module 50 then, in step ST185, checks whether or not the whole of the main body of the OS has been loaded, that is, whether or not all the functional modules 16 to 22 have been loaded into the memory 2. If all the functional modules 16 to 22 have not been loaded into the memory 2 yet, the OS loading and decompression processing module 50 returns to step ST181 in which it continues to load the remaining functional modules of the OS main body.

Settsu, 14:44-52

See also Settsu, 16:7-17:62 and Figs. 1-4, 6-9, 13-14, 20 & 35-36.

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Settsu, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Settsu, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Settsu, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Settsu, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 1.2, 1.3, and 1.4 above.	

7.2 a cache memory device; and;	Settsu, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Settsu, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Settsu, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Settsu, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Settsu, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Settsu discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> The present invention is made to overcome the above problems. It is therefore an object of the present invention to provide an information processing apparatus and a method capable of reducing the time required for booting up itself when it is powered on, and also reducing the time required to start execution of applications to be started automatically when the information processing apparatus is booted up.

Settsu, 1:44-50

> In accordance with another preferred embodiment of the present invention, the OS loading processing module of the mini OS module is an application (AP) execution and OS loading processing module for starting execution of at least a predetermined application module which is located in the file system and which can automatically be started and

run on the operating system when booting up the information processing apparatus, and for loading each of the plurality of functional modules into the memory. Further, the predetermined application module includes a function definition file in which some functional modules required for the application module to run on the operating system are listed. After the mini OS module is loaded into the memory, the mini OS module initializes the mini kernel module and the boot device driver module and then generates and starts execution of a thread for the AP execution and OS loading processing module. After the thread for the AP execution and OS loading processing module is started, the AP execution and OS loading processing module loads the application nodule from the file system into the memory and further loads some functional modules required for the application module into the memory according to the function definition file included in the application module, and then generates and starts execution of a thread for the OS initialization module. After the thread for the OS initialization module is started, the OS initialization module then initializes each of the some functional modules loaded into the memory. And, after the initialization of all of the some functional modules is completed, the application execution and OS loading processing module further loads the remainder of all functional modules included in the OS main body module into the memory and initializes the remainder using the OS initialization processing module while starting execution of the application module as a process.

Settsu, 3:48-4:14

Settsu, Fig. 18.

9.1 maintaining a list of application data associated with an application program;	Settsu, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Settsu, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Settsu, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Settsu discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> The present invention is made to overcome the above problems. It is therefore an object of the present invention to provide an information processing apparatus and a method capable of reducing the time required for booting up itself when it is powered on, and also reducing the time required to start execution of applications to be started automatically when the information processing apparatus is booted up.

Settsu, 1:44-50

> In accordance with another preferred embodiment of the present invention, the OS loading processing module of the mini OS module is an application (AP) execution and OS loading processing module for starting execution of at least a predetermined application module which is located in the file system and which can automatically be started and run on the operating system when booting up the information processing apparatus, and for loading each of the plurality of functional modules

into the memory. Further, the predetermined application module includes a function definition file in which some functional modules required for the application module to run on the operating system are listed. After the mini OS module is loaded into the memory, the mini OS module initializes the mini kernel module and the boot device driver module and then generates and starts execution of a thread for the AP execution and OS loading processing module. After the thread for the AP execution and OS loading processing module is started, the AP execution and OS loading processing module loads the application nodule from the file system into the memory and further loads some functional modules required for the application module into the memory according to the function definition file included in the application module, and then generates and starts execution of a thread for the OS initialization module. After the thread for the OS initialization module is started, the OS initialization module then initializes each of the some functional modules loaded into the memory. And, after the initialization of all of the some functional modules is completed, the application execution and OS loading processing module further loads the remainder of all functional modules included in the OS main body module into the memory and initializes the remainder using the OS initialization processing module while starting execution of the application module as a process.

Settsu, 3:48-4:14

Settsu, Fig. 18.

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Settsu, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

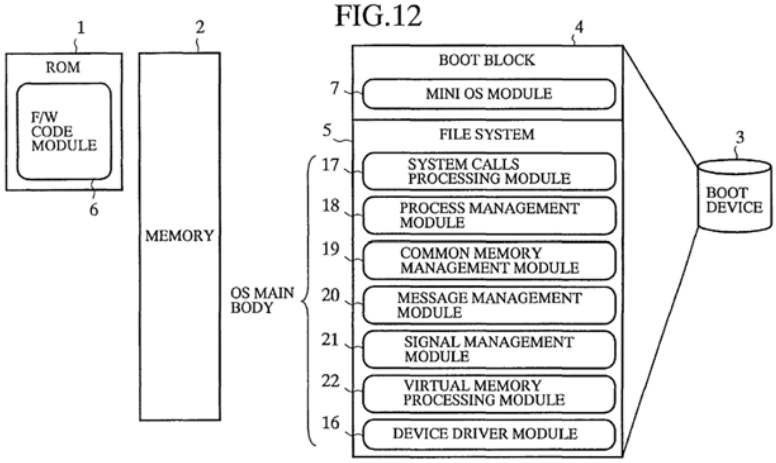
Settsu discloses this limitation:

> Referring next to FIG. 1, there is illustrated a block diagram showing the structure of an information processing apparatus according to a first embodiment of th present invention. In the figure, reference numeral 1 denotes a ROM of the information processing apparatus, 2 denotes a memory of the information processing apparatus, 3 denotes 5 a boot device of the information processing apparatus, 4 denotes a boot block in the boot device 3, 5 denotes a file system I the boot device 3, and 6 denotes a firmware or FI/W code module stored in the ROM 1. The FI/W code module 6 is directly executed on the ROM 1 so as to load data from the boot block 4 in the boot device 3 into the memory 2 and then assume that the loaded data is a code and execute the code after setting up and running diagnostic checks on a hardware or H/W register. Further, reference numeral 7 denotes a mini operating system (OS) module complied and liked in the same way as ordinary program files and located in the boot block 4 within the boot device, the mini OS module having OS functions required for bootstrap processing, and 8 denotes an OS main body module located in the file system 5 within the boot device and provided with OS functions except the OS functions included in the mini OS module 7. When the information processing apparatus is powered on, it goes through initialization and transfers control to the F/W code module 6 stored in the ROM 1.
>
> Referring next to FIG. 2, there is illustrated a diagram showing the structure of the mini OS module 7 stored in the boot block of the boot device of the information processing apparatus according to the first embodiment of the present invention. As shown in the figure, the mini OS module 7 consists of a mini kernel module 9 which is a basic part of

the OS, a boot device driver module 10 for performing I/O operations on the boot device 3, and an OS loading and initialization processing module 11 for loading the OS main body module 8 from the boot device 3 into the memory 2 and for executing the initialization of the OS main body module 8.

Settsu, 7:65-8:35

FIG.12

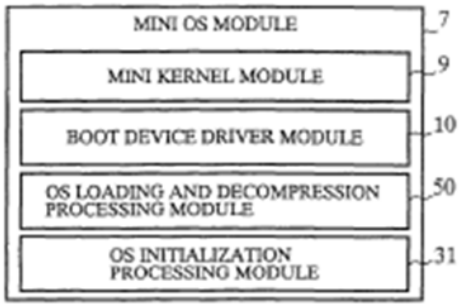


Settsu, Fig. 12

Referring next to FIG. 12, there is illustrated a block diagram showing the structure of an information processing apparatus according to a fourth embodiment of the present invention. In the figure, the same reference numerals as shown in FIG. 5 designate the same or like elements, and therefore the description of those elements will be omitted hereinafter. Like the OS of the second embodiment, the OS of the second embodiment is divided into a mini OS module 7 and a main body of the OS, and the main body is further divided into a plurality of functional modules, such as a system call processing module 17, a process management module 18, a common memory management module 19, a message management module 20, a signal management module 21, a virtual memory processing module 22, and a device driver module 16. The plurality of functional modules are separately stored as compressed files in a file system 5 of a boot device 3.

Settsu, 13:55-65

FIG.13



Settsu, Fig. 13.

> Referring next to FIG. 13, there is illustrated a block diagram showing the structure of the mini OS module 7 of the information processing apparatus according to the fourth embodiment of the present invention. Like the mini OS module 7 of the second embodiment as shown in FIG. 6, the mini OS module 7 of the fourth embodiment is provided with a mini kernel module 9, a boot device driver module 10, and an OS initialization processing module 31. The mini OS module 7 of the fourth embodiment further comprises an OS loading and decompression processing module 50 having a function of decompressing a loaded functional module in addition to the function of the OS load processing module 30 of the second embodiment, instead of the OS load processing module 30.

Settsu, 13:66-14:12

See also Settsu, 16:7-17:62, and Figs. 1-4, 6-9, 14, 20 & 35-36.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Settsu, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Settsu discloses this limitation:

> A method of booting up an information processing apparatus is provided. An operating system is divided into a mini operating system (OS) module having a function of bootstrap and an OS main body module having functions other than the function of bootstrap. The mini OS module can be located in a boot block of a boot device, whereas the OS main body module can be located in a file system of the boot device. A firmware or F/W code module stored in a ROM loads the mini OS module into memory when booting up the information processing apparatus. The mini OS module then loads the OS main body module into memory and then initializes the OS main body module.

Settsu, Abstract

> The present invention relates to an information processing apparatus capable of reducing the time required for booting itself when it is powered on, and a method of booting an information processing apparatus at a high speed.

Settsu, 1:8-12.

> In accordance with an aspect of the present invention, there is provided an information processing apparatus comprising: a boot device divided into a boot block in which a mini operating system (OS) module having a function required for bootstrap processing is located and a file system in which an operating system (OS) main body module having functions other than the function of bootstrap. . . .

Settsu, 1:51-57

> In accordance with another preferred embodiment of the present invention, the plurality of functional modules, into which the OS main

body module is divided, are stored as compressed files in the file system and the loading and initialization processing module of the mini OS module is divided into an OS loading and decompression processing module and an OS initialization module. Further, the mini OS module generates and starts execution of a thread for the OS loading and decompression processing module after the mini OS module initializes the mini kernel module and the boot device driver module. After the thread for the OS loading and decompression processing module is started, the OS loading and decompression processing module loads each of the plurality of functional modules into the memory and decompresses the loaded functional module, and then generates and starts execution of a thread for the OS initialization module. After the thread for the OS initialization module is executed, the OS initialization module initializes each of the plurality of functional modules loaded into the memory and decompressed.

Settsu, 3:6-25

Referring next to FIG. 1, there is illustrated a block diagram showing the structure of an information processing apparatus according to a first embodiment of th present invention. In the figure, reference numeral 1 denotes a ROM of the information processing apparatus, 2 denotes a memory of the information processing apparatus, 3 denotes 5 a boot device of the information processing apparatus, 4 denotes a boot block in the boot device 3, 5 denotes a file system I the boot device 3, and 6 denotes a firmware or FI/W code module stored in the ROM 1. The FI/W code module 6 is directly executed on the ROM 1 so as to load data from the boot block 4 in the boot device 3 into the memory 2 and then assume that the loaded data is a code and execute the code after setting up and running diagnostic checks on a hardware or H/W register. Further, reference numeral 7 denotes a mini operating system (OS) module complied and liked in the same way as ordinary program files and located in the boot block 4 within the boot device, the mini OS module having OS functions required for bootstrap processing, and 8 denotes an OS main body module located in the file system 5 within the boot device and provided with OS functions except the OS functions included in the mini OS module 7. When the information processing apparatus is powered on, it goes through initialization and transfers control to the F/W code module 6 stored in the ROM 1.

Referring next to FIG. 2, there is illustrated a diagram showing the structure of the mini OS module 7 stored in the boot block of the boot device of the information processing apparatus according to the first embodiment of the present invention. As shown in the figure, the mini OS module 7 consists of a mini kernel module 9 which is a basic part of

the OS, a boot device driver module 10 for performing I/O operations on the boot device 3, and an OS loading and initialization processing module 11 for loading the OS main body module 8 from the boot device 3 into the memory 2 and for executing the initialization of the OS main body module 8.

Settsu, 7:65-8:35

Referring next to FIG. 3, there is illustrated a diagram showing the structure of the OS main body module 8 of the information processing apparatus according to the first embodiment of the present invention. As shown in the figure, the OS main body module 8 consists of a kernel module 15 having kernel functions except the functions included in the mini kernel module 9 of FIG. 2, and a device driver module 16 for performing I/O operations on devices (not shown) except the boot device 3. . . . The OS main body module 8 is located within the file system 5 of the boot device 3 and is loaded into the memory 2 by the mini OS module 7. The OS main body module 8 then goes through initialization.

Settsu, 8:47-9:3

Referring next to FIG. 4, there is illustrated a flow chart showing operations of the mini OS module 7 of the information processing apparatus according to the first embodiment of the present invention. The F/W code module 6 loads the mini OS module 7 into the memory 2 and transfers control to the mini OS module 7. The mini OS module 7 then, in step ST101, executes initialization of the mini kernel module 9. In this case, the thread management module 12, the I/O management module 13, and the thread communication management module 14 are initialized and their functions are available now. Next, the mini OS module 7, in step ST102, executes initialization of the boot device driver module 10. The boot device driver module 10 registers an interrupt request from the boot device into an interrupt table of the I/O management module 13 so as to handle the interrupt request, and generates and starts execution of a thread for boot device I/O processing by using the thread management module 12.

Settsu, 9:7-21.

The mini OS module 7, in step ST103, generates a thread for the OS loading and initialization processing module 11 by using the thread management module 12. The mini OS module 7 further, in step ST104, starts execution of the thread by using the thread management module 12. The OS loading and initialization processing module 11 is thus

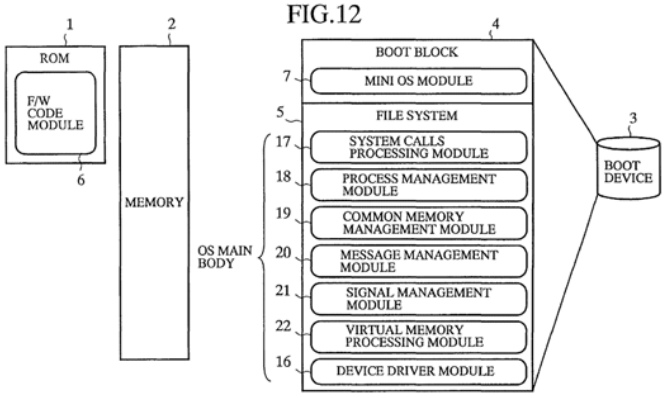
started up as the thread. After that, the mini OS module 7 transfers control to the OS loading and initialization processing module 11.

Settsu, 9:30-39.

As previously mentioned, in accordance with the second embodiment of the present invention, the main body of the OS is divided into a plurality of functional modules according a plurality of functions to be performed by the main body of the OS. Further, the plurality of functional modules are separately stored in the file system 5. In addition, the OS load processing and the OS initialization processing can be performed in parallel with each other after any one of the plurality of functional modules of the OS main body is loaded into the memory. As a result, while the CPU waits for the occurrence of an event in performing the OS load or initialization processing, the CPU does not idle but the CPU performs another processing. Accordingly, the second embodiment of the present invention provides an advantage of being able to further reduce the time required for booting up the information processing apparatus.

Settsu, 12:1-16.

FIG.12



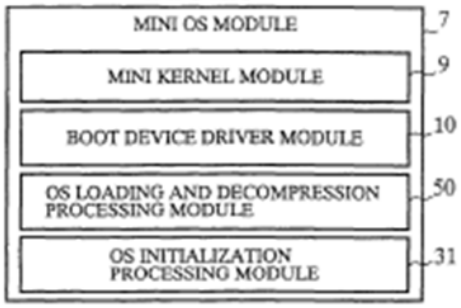
Settsu, Fig. 12

Referring next to FIG. 12, there is illustrated a block diagram showing the structure of an information processing apparatus according to a fourth embodiment of the present invention. In the figure, the same reference numerals as shown in FIG. 5 designate the same or like elements, and therefore the description of those elements will be omitted hereinafter. Like the OS of the second embodiment, the OS of the second embodiment is divided into a mini OS module 7 and a main body of the OS, and the main body is further divided into a plurality of functional modules, such as a system call processing module 17, a

process management module 18, a common memory management module 19, a message management module 20, a signal management module 21, a virtual memory processing module 22, and a device driver module 16. The plurality of functional modules are separately stored as compressed files in a file system 5 of a boot device 3.

Settsu, 13:55-65

FIG.13



Settsu, Fig. 13.

Referring next to FIG. 13, there is illustrated a block diagram showing the structure of the mini OS module 7 of the information processing apparatus according to the fourth embodiment of the present invention. Like the mini OS module 7 of the second embodiment as shown in FIG. 6, the mini OS module 7 of the fourth embodiment is provided with a mini kernel module 9, a boot device driver module 10, and an OS initialization processing module 31. The mini OS module 7 of the fourth embodiment further comprises an OS loading and decompression processing module 50 having a function of decompressing a loaded functional module in addition to the function of the OS load processing module 30 of the second embodiment, instead of the OS load processing module 30.

Settsu, 13:66-14:12

Referring next to FIG. 14, there is illustrated a flow chart showing operations of the OS loading and decompression processing module 50 of the information processing apparatus according to the fourth embodiment of the present invention. The OS loading and decompression processing module 50 to which control from the mini OS module 7 has been transferred, in step ST181, loads the main body of the OS stored in the file system 5 of the boot device 3, such as the system call processing module 17, the process management module 18, the common memory management module 19, the message

> management module 20, the signal management module 21, the virtual memory processing module 22, and the device driver module 16, into the memory 2. In performing step ST181, the OS loading and decompression processing module 50 loads any one of those functional modules 16 to 22 first. The OS loading and decompression processing module 50 then, in step ST182, decompresses one functional module loaded into the memory. Since the plurality of functional modules 16 to 22 are stored as compresses files in the file system 5 of the boot device, the functional module loaded into the memory 2 is compressed data. Therefore, in performing step ST182, the OS loading and decompression processing module 50 decompresses the compressed data so as to convert it into executable code and data.

Settsu, 14:13-37

> As previously mentioned, in accordance with the fourth embodiment of the present invention, the main body of the OS is divided into a plurality of functional modules according a plurality of functions to be performed by the main body, and the plurality of functional modules are stored as compressed files in the file system **5** of the boot device. Further, the OS loading and decompression processing module **50** decompresses each of the plurality of functional modules each time it loads each of them into memory. As a result, the time required for I/O processing can be reduced. Accordingly, the fourth embodiment of the present invention provides an advantage of being able to further reduce the time required for booting up the information processing apparatus.

Settsu, 14:58-15:5

See also Settsu, 8:21-35, 8:66-9:11, 11:7-9, 11:18-39, 13:49-15:5, 16:7-17:62, and Figs. 1-4, 6-9, 14, 20 & 35-36.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Settsu, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Settsu, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Settsu, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Settsu discloses this limitation:

See Claims 1.3 and 1.4 above.

See also

> In accordance with another preferred embodiment of the present invention, the plurality of functional modules, into which the OS main body module is divided, are stored as compressed files in the file system and the loading and initialization processing module of the mini OS module is divided into an OS loading and decompression processing module and an OS initialization module. Further, the mini OS module generates and starts execution of a thread for the OS loading and decompression processing module after the mini OS module initializes the mini kernel module and the boot device driver module. After the thread for the OS loading and decompression processing module is started, the OS loading and decompression processing module loads each of the plurality of functional modules into the memory and decompresses the loaded functional module, and then generates and starts execution of a thread for the OS initialization module. After the thread for the OS initialization module is executed, the OS initialization module initializes each of the plurality of functional modules loaded into the memory and decompressed.

Settsu, 3:6-25

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Settsu, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Settsu discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

See also

> In accordance with another preferred embodiment of the present invention, the plurality of functional modules, into which the OS main body module is divided, are stored as compressed files in the file system and the loading and initialization processing module of the mini OS module is divided into an OS loading and decompression processing module and an OS initialization module. Further, the mini OS module generates and starts execution of a thread for the OS loading and decompression processing module after the mini OS module initializes the mini kernel module and the boot device driver module. After the thread for the OS loading and decompression processing module is started, the OS loading and decompression processing module loads each of the plurality of functional modules into the memory and decompresses the loaded functional module, and then generates and starts execution of a thread for the OS initialization module. After the thread for the OS initialization module is executed, the OS initialization module initializes each of the plurality of functional modules loaded into the memory and decompressed.

Settsu, 3:6-25

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Settsu, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Settsu, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Settsu, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Settsu, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Settsu, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Settsu, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Settsu, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Settsu, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Settsu, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Settsu, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Settsu, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Settsu, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Settsu, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Settsu, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Settsu, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Settsu, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Settsu, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Settsu, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 15 and 27 above	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Settsu, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Settsu discloses this limitation: *See* Claims 16 and 27 above	

Appendix A18
Invalidity of U.S. Patent 7,181,608 based on Shinjo

U.S. Patent No. 5,269,022 to Shinjo ("Shinjo") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A18
Invalidity of U.S. Patent 7,181,608 based on Shinjo

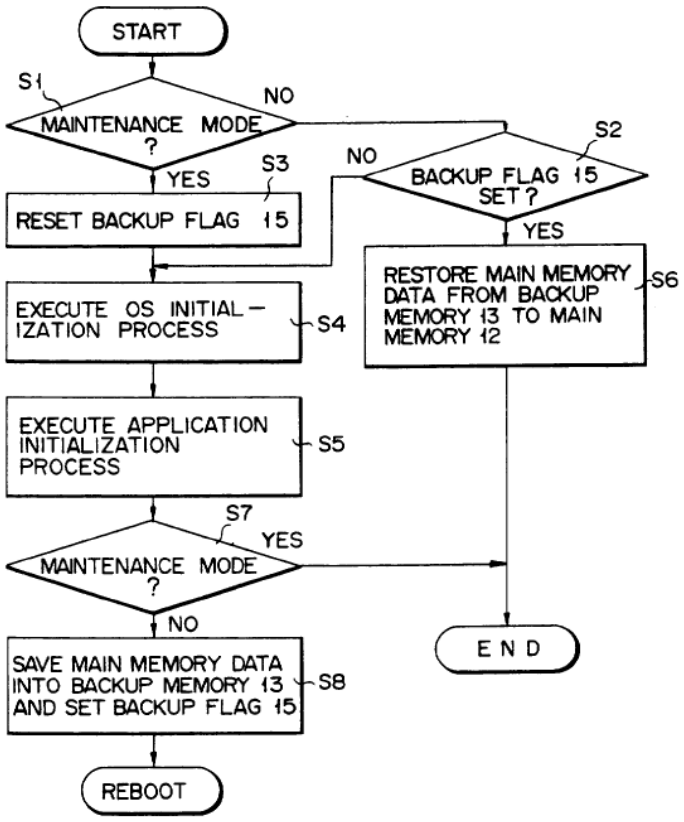
1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Shinjo, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shinjo discloses this limitation:

> In a computer system, when the system is first booted in a normal mode, main memory data stored in a main memory immediately after the system is booted, is stored as backup data in a backup memory or the like. A backup flag representing whether or not the backup data can be restored is set and the system is rebooted. When the system is next booted in the normal mode, the backup data stored in the backup memory or the like is restored as the main memory data in the main memory. The backup flag is automatically reset in a maintenance mode.

Shinjo, Abstract



F I G. 2

Shinjo, Figure 2

> The above-described process requires long time since the number of times of access to a recording medium such as a disk is very large and an overhead for retrieval of file, production of process or the like is increased. In other words, boot time required until an operation as a computer system starts, is lengthened. It is thus desirable to achieve an apparatus capable of booting a computer system at high speed.

Shinjo, 1:22-29

> It is an object of the present invention to provide a method and an apparatus for booting a computer system.

Shinjo, 1:32-34

> According to one aspect of the present invention, there is provided a method for booting a computer system, the method comprising the steps of: setting a boot mode for booting the computer system; determining whether or not the set boot mode is a normal mode; determining whether or not a flag is set when the boot mode is the normal mode, the flag representing whether or not backup data is

restorable; storing main memory data to be stored in a main memory immediately after the computer system is booted, as the backup data, into a backup memory when the flag is reset; and restoring the backup data stored in the backup memory, as the main memory data, into the main memory when the flag is set.
Shinjo, 1:32-48

The boot mode setting unit 17 is constituted of, for example, a service processor (SVP) and used to set a boot mode in the computer system. The boot mode includes a maintenance mode for software maintenance such as replacement of the programs and patch and a mode (normal mode) other than the maintenance mode. The normal mode includes a quick start mode in which a high speed boot can be executed using the backup data and a saving mode in which the main memory data stored in the main memory 12 is saved in the backup memory 13 as the backup data, immediately after a normal boot is executed. It depends upon the set/reset state of the backup flag 15 which of the quick start mode and the saving mode is selected. More specifically, in the normal mode, when the backup flag 15 is set, the quick start mode is selected and, when the backup flag 15 is reset, the saving mode is selected.
Shinjo, 2:51-2:68

Since the backup flag 15 has been set, the boot process is started in the quick start mode through steps S1 and S2 after the reboot. That is, in step S6, the backup data stored in the backup memory 13 is restored into the main memory 12 as the main memory data. Since the computer system is thus restored to the state immediately after the boot process and the running environment is set, the boot process of the computer system can be completed without executing the OS initialization process of step S3 or the application initialization process of step S4.
Shinjo, 3:35-45

Therefore, the boot process of the present invention can be executed at higher speed than the conventional boot process.
Shinjo, 3:62-64

As described above, according to the present invention, when the system is first booted, the saving mode is selected. Immediately after the system is booted, the main memory data stored in the main memory is saved as backup data into the backup memory (backup file), and the backup flag is set. The system is then rebooted. Therefore, when the system is next booted in the normal mode, the quick start mode is selected and the backup data saved into the backup memory (backup file) is restored as the main memory data in the main memory. The boot process of the system is completed only by the above process, and the state immediately after the system is booted is restored.
Shinjo, 4:45-57

> In the conventional system, a very large number of times of disk access are necessary for the OS initialization process such as setting of the running environment of an OS and for the application initialization process such as setting of the running environment of an application program, and long time is required for the boot process. In the present invention, however, the system can be booted at high speed.
>
> Shinjo, 4:58-65

1.1 maintaining a list of boot data used for booting a computer system;	Shinjo, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shinjo discloses this limitation:

> The present invention relates to a method and an apparatus for booting a computer system.

Shinjo, 1:9-10

> In a computer system, generally, whenever the system is booted, a boot process for loading firmware, an initial program loader (IPL) program and an initialize (INZ) program, an initial program loader process, and an initialization process are executed.

Shinjo, 1:11-16

> According to one aspect of the present invention, there is provided a method for booting a computer system, the method comprising the steps of: setting a boot mode for booting the computer system; determining whether or not the set boot mode is a normal mode; determining whether or not a flag is set when the boot mode is the normal mode, the flag representing whether or not backup data is restorable; storing main memory data to be stored in a main memory immediately after the computer system is booted, as the backup data, into a backup memory when the flag is reset; and restoring the backup data stored in the backup memory, as the main memory data, into the main memory when the flag is set.

Shinjo, 1:35-48

> According to another aspect of the present invention, there is provided an apparatus for booting a computer system, the apparatus comprising: a main memory for storing main memory data; means for setting a boot mode for booting the computer system; determining means for determining whether or not the boot mode is a normal mode; a flag to be set/reset in accordance with a determination result by the determining means; and a backup memory for storing the main memory data to be stored in the main memory immediately after the computer system is booted, as backup data when the boot mode is the normal mode and the flag is reset, and wherein the backup data stored in the

backup memory is restored as the main memory data into the main memory when the boot mode is the normal mode and the flag is set.
Shinjo, 1:49-64

The boot mode setting unit 17 is constituted of, for example, a service processor (SVP) and used to set a boot mode in the computer system. The boot mode includes a maintenance mode for software maintenance such as replacement of the programs and patch and a mode (normal mode) other than the maintenance mode. The normal mode includes a quick start mode in which a high speed boot can be executed using the backup data and a saving mode in which the main memory data stored in the main memory 12 is saved in the backup memory 13 as the backup data, immediately after a normal boot is executed. It depends upon the set/reset state of the backup flag 15 which of the quick start mode and the saving mode is selected. More specifically, in the normal mode, when the backup flag 15 is set, the quick start mode is selected and, when the backup flag 15 is reset, the saving mode is selected.
Shinjo, 2:51-68

When a boot command is output from the boot mode setting unit 17 to the CPU 11, an operation using the initial program loader (IPL) program is started by the CPU 11.
Shinjo, 3:4-7

In step S1, it is determined whether or not the boot mode designated by the boot command output from the boot mode setting unit 17 is the maintenance mode. If the boot mode is not the maintenance mode, i.e., if it is the normal mode, it is determined whether or not the backup flag 15 of the backup memory 13 is set (step S2). That is, it is determined whether a boot process is executed in the quick start mode or saving mode.
Shinjo, 3:8-15

In step S2, when the backup flag 15 is reset, it is determined that the backup data stored in the backup memory 13 cannot be restored. Since this determination is made in the first boot process, the same boot process as a conventional one is executed. In other words, the operating system (OS) initialization process and the application initialization process are executed by the initialization program (steps S4 and S5).
Shinjo, 3:16-23

Since the backup flag 15 has been set, the boot process is started in the quick start mode through steps S1 and S2 after the reboot. That is, in step S6, the backup data stored in the backup memory 13 is restored into the main memory 12 as the main memory data. Since the computer system is thus restored to the state immediately after the boot process and the running environment is set, the boot process of the computer system can be completed without executing the

> OS initialization process of step S3 or the application initialization process of step S4.
>
> Shinjo, 3:35-45
>
> In the computer system according to the first embodiment wherein the main memory data stored in the main memory 12 is saved into the backup memory 13 as the backup data, when the system power source is turned off, the backup data is erased. It is thus necessary to boot the system in the same manner as the conventional apparatus and save the main memory data stored in the main memory 12 into the backup memory 13 as the backup data immediately after the system is booted. By backing up the backup memory 13 by a battery or the like, such boot process in the system is executed only at the first time.
>
> Shinjo, 3:65-4:8
>
> As described above, according to the present invention, when the system is first booted, the saving mode is selected. Immediately after the system is booted, the main memory data stored in the main memory is saved as backup data into the backup memory (backup file), and the backup flag is set. The system is then rebooted. Therefore, when the system is next booted in the normal mode, the quick start mode is selected and the backup data saved into the backup memory (backup file) is restored as the main memory data in the main memory. The boot process of the system is completed only by the above process, and the state immediately after the system is booted is restored.
>
> Shinjo, 4:45-67

1.2 initializing a central processing unit of the computer system;	Shinjo, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shinjo discloses this limitation:

> The CPU 11 controls the entire computer system.

Shinjo, 2:32

> When a boot command is output from the boot mode setting unit 17 to the CPU 11, an operation using the initial program loader (IPL) program is started by the CPU 11.

Shinjo, 3:4-7

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Shinjo, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shinjo discloses this limitation:

> In the initialization process, a resident load module is loaded into a main memory, various control blocks are produced, the running environment of an operating system (OS) is set, and the running environment of an application system is set (for example, a control process is produced).

Shinjo, 1:16-21

> According to one aspect of the present invention, there is provided a method for booting a computer system, the method comprising the steps of: setting a boot mode for booting the computer system; determining whether or not the set boot mode is a normal mode; determining whether or not a flag is set when the boot mode is the normal mode, the flag representing whether or not backup data is restorable; storing main memory data to be stored in a main memory immediately after the computer system is booted, as the backup data, into a backup memory when the flag is reset; and restoring the backup data stored in the backup memory, as the main memory data, into the main memory when the flag is set.

Shinjo, 1:35-48

> In step S7, when the designated boot mode is not the maintenance mode, i.e., when it is the normal mode, the saving mode is selected, and the main memory data stored in the main memory 12 is saved into the backup memory 13 as the backup data and the backup flag 15 of the backup memory 13 is set (step S8). The process of step S8 is completed and then a reboot is executed.

Shinjo, 3:28-34

> Since the backup flag 15 has been set, the boot process is started in the quick start mode through steps S1 and S2 after the reboot. That is, in step S6, the backup data stored in the backup memory 13 is restored into the main memory 12 as the main memory data. Since the computer system is thus restored to the state immediately after the boot process and the running environment is set, the boot process of the computer system can be completed without executing the OS initialization process of step S3 or the application initialization process of step S4.
>
> Shinjo, 3:35-45

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Shinjo, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shinjo discloses this limitation:

> When a boot command is output from the boot mode setting unit 17 to the CPU 11, an operation using the initial program loader (IPL) program is started by the CPU 11.

Shinjo, 3:4-7

> In step S1, it is determined whether or not the boot mode designated by the boot command output from the boot mode setting unit 17 is the maintenance mode. If the boot mode is not the maintenance mode, i.e., if it is the normal mode, it is determined whether or not the backup flag 15 of the backup memory 13 is set (step S2). That is, it is determined whether a boot process is executed in the quick start mode or saving mode.

Shinjo, 3:8-15

> Since the backup flag 15 has been set, the boot process is started in the quick start mode through steps S1 and S2 after the reboot. That is, in step S6, the backup data stored in the backup memory 13 is restored into the main memory 12 as the main memory data. Since the computer system is thus restored to the state immediately after the boot process and the running environment is set, the boot process of the computer system can be completed without executing the

OS initialization process of step S3 or the application initialization process of step S4. Shinjo, 3:35-45

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Shinjo, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shinjo discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

> The boot mode setting unit 17 is constituted of, for example, a service processor (SVP) and used to set a boot mode in the computer system. The boot mode includes a maintenance mode for software maintenance such as replacement of the programs and patch and a mode (normal mode) other than the maintenance mode. The normal mode includes a quick start mode in which a high speed boot can be executed using the backup data and a saving mode in which the main memory data stored in the main memory 12 is saved in the backup memory 13 as the backup data, immediately after a normal boot is executed. It depends upon the set/reset state of the backup flag 15 which of the quick start mode and the saving mode is selected. More specifically, in the normal mode, when the backup flag 15 is set, the quick start mode is selected and, when the backup flag 15 is reset, the saving mode is selected.

Shinjo, 2:51-2:68

> After the backup flag 15 is reset in step S3, the boot process is executed as in the conventional system, i.e., the OS initialization process and the application initialization process are executed (steps S4 and S5).

Shinjo, 3:54-57

> In the conventional system, a very large number of times of disk access are necessary for the OS initialization process such as setting of the running environment of an OS and for the application initialization process such as setting of the running environment of an application program, and long time is required for the boot process. In the present invention, however, the system can be booted at high speed.

Shinjo, 4:58-65

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Shinjo, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shinjo discloses this limitation:

See Claims 1.3, and 1.4 above.

> In the initialization process, a resident load module is loaded into a main memory, various control blocks are produced, the running environment of an operating system (OS) is set, and the running environment of an application system is set (for example, a control process is produced).

Shinjo, 1:16-21

4. The method of claim 1, further comprising updating the list of boot data.	Shinjo, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shinjo discloses this limitation:

See Claim 1.1 above.

> The boot mode setting unit 17 is constituted of, for example, a service processor (SVP) and used to set a boot mode in the computer system. The boot mode includes a maintenance mode for software maintenance such as replacement of the programs and patch and a mode (normal mode) other than the maintenance mode. The normal mode includes a quick start mode in which a high speed boot can be executed using the backup data and a saving mode in which the main memory data stored in the main memory 12 is saved in the backup memory 13 as the backup data, immediately after a normal boot is executed. It depends upon the set/reset state of the backup flag 15 which of the quick start mode and the saving mode is selected. More specifically, in the normal mode, when the backup flag 15 is set, the quick start mode is selected and, when the backup flag 15 is reset, the saving mode is selected.

Shinjo, 2:52-68

> When software maintenance such as replacement of programs and patch is performed, the boot process is executed in the maintenance mode and thus the backup data cannot be automatically restored. When the system is next booted, the saving mode is selected again. Therefore, the update backup data can always be stored in the backup memory (backup file).

Shinjo, 5:3-9

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Shinjo, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Shinjo, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Shinjo, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Shinjo, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shinjo discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

> In the initialization process, a resident load module is loaded into a main memory, various control blocks are produced, the running environment of an operating system (OS) is set, and the running environment of an application system is set (for example, a control process is produced).

Shinjo, 1:16-21

7.2 a cache memory device; and;	Shinjo, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Shinjo, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Shinjo, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Shinjo, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Shinjo, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shinjo discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

> The boot mode setting unit 17 is constituted of, for example, a service processor (SVP) and used to set a boot mode in the computer system. The boot mode includes a maintenance mode for software maintenance such as replacement of the programs and patch and a mode (normal mode) other than the maintenance mode. The normal mode includes a quick start mode in which a high speed boot can be executed using the backup data and a saving mode in which the main memory data stored in the main memory 12 is saved in the backup memory 13 as the backup data, immediately after a normal boot is executed. It depends upon the set/reset state of the backup flag 15 which of the quick start mode and the saving mode is selected. More specifically, in the normal mode, when the backup flag 15 is set, the quick start mode is selected and, when the backup flag 15 is reset, the saving mode is selected.

Shinjo, 2:51-2:68

> After the backup flag 15 is reset in step S3, the boot process is executed as in the conventional system, i.e., the OS initialization process and the application

initialization process are executed (steps S4 and S5).
Shinjo, 3:54-57

> In the conventional system, a very large number of times of disk access are necessary for the OS initialization process such as setting of the running environment of an OS and for the application initialization process such as setting of the running environment of an application program, and long time is required for the boot process. In the present invention, however, the system can be booted at high speed.

Shinjo, 4:58-65

9.1 maintaining a list of application data associated with an application program;	Shinjo, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Shinjo, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Shinjo, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shinjo discloses this limitation:

See Claims 1.4, 2, and 8 above.

> The boot mode setting unit 17 is constituted of, for example, a service processor (SVP) and used to set a boot mode in the computer system. The boot mode includes a maintenance mode for software maintenance such as replacement of the programs and patch and a mode (normal mode) other than the maintenance mode. The normal mode includes a quick start mode in which a high speed boot can be executed using the backup data and a saving mode in which the main memory data stored in the main memory 12 is saved in the backup memory 13 as the backup data, immediately after a normal boot is executed. It depends upon the set/reset state of the backup flag 15 which of the quick start mode and the saving mode is selected. More specifically, in the normal mode, when the backup flag 15 is set, the quick start mode is selected and, when the backup flag 15 is reset, the saving mode is selected.

Shinjo, 2:51-2:68

> After the backup flag 15 is reset in step S3, the boot process is executed as in the conventional system, i.e., the OS initialization process and the application initialization process are executed (steps S4 and S5).

Shinjo, 3:54-57

Appendix A18
Invalidity of U.S. Patent 7,181,608 based on Shinjo

> In the conventional system, a very large number of times of disk access are necessary for the OS initialization process such as setting of the running environment of an OS and for the application initialization process such as setting of the running environment of an application program, and long time is required for the boot process. In the present invention, however, the system can be booted at high speed.
>
> Shinjo, 4:58-65

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Shinjo, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shinjo discloses this limitation:

> The present invention relates to a method and an apparatus for booting a computer system.

Shinjo, 1:9-10

> In a computer system, generally, whenever the system is booted, a boot process for loading firmware, an initial program loader (IPL) program and an initialize (INZ) program, an initial program loader process, and an initialization process are executed.

Shinjo, 1:11-16

> According to one aspect of the present invention, there is provided a method for booting a computer system, the method comprising the steps of: setting a boot mode for booting the computer system; determining whether or not the set boot mode is a normal mode; determining whether or not a flag is set when the boot mode is the normal mode, the flag representing whether or not backup data is restorable; storing main memory data to be stored in a main memory immediately after the computer system is booted, as the backup data, into a backup memory when the flag is reset; and restoring the backup data stored in the backup memory, as the main memory data, into the main memory when the flag is set.

Shinjo, 1:35-48

> According to another aspect of the present invention, there is provided an apparatus for booting a computer system, the apparatus comprising: a main memory for storing main memory data; means for setting a boot mode for booting the computer system; determining means for determining whether or not the boot mode is a normal mode; a flag to be set/reset in accordance with a

determination result by the determining means; and a backup memory for storing the main memory data to be stored in the main memory immediately after the computer system is booted, as backup data when the boot mode is the normal mode and the flag is reset, and wherein the backup data stored in the backup memory is restored as the main memory data into the main memory when the boot mode is the normal mode and the flag is set.

Shinjo, 1:49-64

The boot mode setting unit 17 is constituted of, for example, a service processor (SVP) and used to set a boot mode in the computer system. The boot mode includes a maintenance mode for software maintenance such as replacement of the programs and patch and a mode (normal mode) other than the maintenance mode. The normal mode includes a quick start mode in which a high speed boot can be executed using the backup data and a saving mode in which the main memory data stored in the main memory 12 is saved in the backup memory 13 as the backup data, immediately after a normal boot is executed. It depends upon the set/reset state of the backup flag 15 which of the quick start mode and the saving mode is selected. More specifically, in the normal mode, when the backup flag 15 is set, the quick start mode is selected and, when the backup flag 15 is reset, the saving mode is selected.

Shinjo, 2:51-68

When a boot command is output from the boot mode setting unit 17 to the CPU 11, an operation using the initial program loader (IPL) program is started by the CPU 11.

Shinjo, 3:4-7

In step S1, it is determined whether or not the boot mode designated by the boot command output from the boot mode setting unit 17 is the maintenance mode. If the boot mode is not the maintenance mode, i.e., if it is the normal mode, it is determined whether or not the backup flag 15 of the backup memory 13 is set (step S2). That is, it is determined whether a boot process is executed in the quick start mode or saving mode.

Shinjo, 3:8-15

In step S2, when the backup flag 15 is reset, it is determined that the backup data stored in the backup memory 13 cannot be restored. Since this determination is made in the first boot process, the same boot process as a conventional one is executed. In other words, the operating system (OS) initialization process and the application initialization process are executed by the initialization program (steps S4 and S5).

Shinjo, 3:16-23

Since the backup flag 15 has been set, the boot process is started in the quick start mode through steps S1 and S2 after the reboot. That is, in step S6, the

backup data stored in the backup memory 13 is restored into the main memory 12 as the main memory data. Since the computer system is thus restored to the state immediately after the boot process and the running environment is set, the boot process of the computer system can be completed without executing the OS initialization process of step S3 or the application initialization process of step S4.

Shinjo, 3:35-45

In the computer system according to the first embodiment wherein the main memory data stored in the main memory 12 is saved into the backup memory 13 as the backup data, when the system power source is turned off, the backup data is erased. It is thus necessary to boot the system in the same manner as the conventional apparatus and save the main memory data stored in the main memory 12 into the backup memory 13 as the backup data immediately after the system is booted. By backing up the backup memory 13 by a battery or the like, such boot process in the system is executed only at the first time.

Shinjo, 3:65-4:8

As described above, according to the present invention, when the system is first booted, the saving mode is selected. Immediately after the system is booted, the main memory data stored in the main memory is saved as backup data into the backup memory (backup file), and the backup flag is set. The system is then rebooted. Therefore, when the system is next booted in the normal mode, the quick start mode is selected and the backup data saved into the backup memory (backup file) is restored as the main memory data in the main memory. The boot process of the system is completed only by the above process, and the state immediately after the system is booted is restored.

Shinjo, 4:45-67

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Shinjo, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shinjo discloses this limitation:

> The boot mode setting unit 17 is constituted of, for example, a service processor (SVP) and used to set a boot mode in the computer system. The boot mode includes a maintenance mode for software maintenance such as replacement of the programs and patch and a mode (normal mode) other than the maintenance mode. The normal mode includes a quick start mode in which a high speed boot can be executed using the backup data and a saving mode in which the main memory data stored in the main memory 12 is saved in the backup memory 13 as the backup data, immediately after a normal boot is executed. It depends upon the set/reset state of the backup flag 15 which of the quick start mode and the saving mode is selected. More specifically, in the normal mode, when the backup flag 15 is set, the quick start mode is selected and, when the backup flag 15 is reset, the saving mode is selected.

Shinjo, 2:51-2:68

> Since the backup flag 15 has been set, the boot process is started in the quick start mode through steps S1 and S2 after the reboot. That is, in step S6, the backup data stored in the backup memory 13 is restored into the main memory 12 as the main memory data. Since the computer system is thus restored to the state immediately after the boot process and the running environment is set, the boot process of the computer system can be completed without executing the OS initialization process of step S3 or the application initialization process of step S4.

Shinjo, 3:35-45

> The boot mode setting unit 17 is constituted of, for example, a service processor (SVP) and used to set a boot mode in the computer system. The boot mode includes a maintenance mode for software maintenance such as replacement of the programs and patch and a mode (normal mode) other than the maintenance mode. The normal mode includes a quick start mode in which a high speed boot can be executed using the backup data and a saving mode in which the main memory data stored in the main memory 12 is saved in the backup memory 13 as the backup data, immediately after a normal boot is

executed. It depends upon the set/reset state of the backup flag 15 which of the quick start mode and the saving mode is selected. More specifically, in the normal mode, when the backup flag 15 is set, the quick start mode is selected and, when the backup flag 15 is reset, the saving mode is selected.

Shinjo, 2:51-2:68

In step S7, when the designated boot mode is not the maintenance mode, i.e., when it is the normal mode, the saving mode is selected, and the main memory data stored in the main memory 12 is saved into the backup memory 13 as the backup data and the backup flag 15 of the backup memory 13 is set (step S8). The process of step S8 is completed and then a reboot is executed.

Shinjo, 3:28-34

In a system according to the third embodiment as shown in FIG. 4, the memory 14 includes the backup memory 13, and the disk unit 19 includes the backup file 23. In the saving mode, therefore, the main memory data stored in the main memory 12 can be saved as backup data into the backup memory 13 and backup file 23, and the backup flags 15 and 25 can be set.

Shinjo, 4:24-31

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Shinjo, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Shinjo, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shinjo discloses this limitation:

See Claims 1.3 and 1.4 above.

> A disk unit 19 of a computer system according to the second embodiment as shown in FIG. 3 can be used in place of the backup memory 13 shown in FIG. 1. The disk unit 19 includes a backup file 23 for storing the main memory data stored in the main memory 12 as backup data and a backup flag 25. Even through the system power source is turned off, the backup data stored in the backup file 23 is not erased. In the system according to the second embodiment, since the backup data as the main memory data is saved and restored between the main memory 12 and disk unit 19, disk access occurs. Therefore, the time required for the boot process of the system according to the first embodiment is longer than the time required for that of the system according to the first embodiment.

Shinjo, 4:9-24

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Shinjo, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Shinjo, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 1.3, 1.4, and 15 above.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Shinjo, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Shinjo, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Shinjo, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Shinjo, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Shinjo, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Shinjo, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Shinjo, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Shinjo, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Shinjo, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Shinjo, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Shinjo, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Shinjo, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Shinjo, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Shinjo, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Shinjo, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Shinjo, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Shinjo, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Shinjo, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 15 and 27 above	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Shinjo, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shinjo discloses this limitation: *See* Claims 16 and 27 above	

Appendix A19
Invalidity of U.S. Patent 7,181,608 based on Shipman

U.S. Patent No. 5,671,413 to Shipman ("Shipman") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A19
Invalidity of U.S. Patent 7,181,608 based on Shipman

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Shipman, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: > The services to be provided by a basic input/output system (BIOS) of a computer system are implemented via a number of independently executable service components. Additionally, the BIOS is provided with a decompression dispatcher for decompressing and dispatching the service components into random access memory (RAM) of the computer system for execution on an as needed basis, and optionally removing the dispatched service components when they are no longer needed. As a result, the service components may be stored in a non-volatile storage in a compressed state, allowing more services to be implemented without requiring more non-volatile storage. Shipman, Abstract	

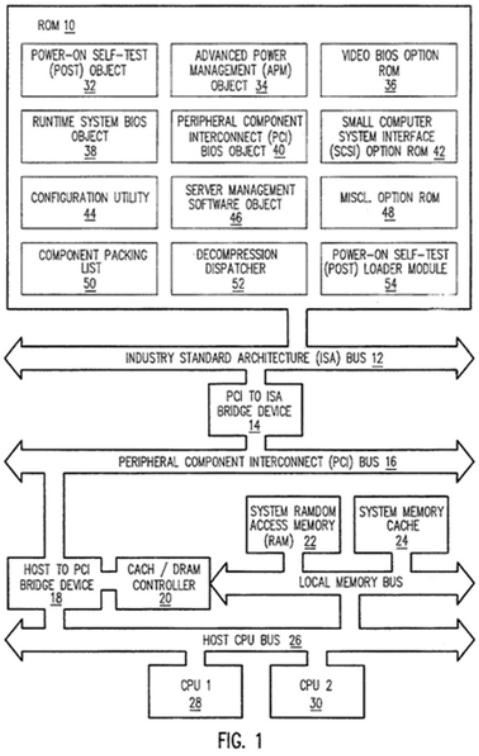


FIG. 1

Shipman, Fig. 1

> In IBM compatible personal computers a set of programs called basic input/output system (BIOS) are encoded in read-only memory (ROM). The BIOS facilitates the transfer of data and instructions between a central processing unit (CPU) and peripheral devices such as disk drives. Computer systems are designed to perform functional tests of the BIOS every time the computer is turned on. When the computer is turned on, BIOS is copied to an area of random access memory (RAM) set aside for it. Since a RAM is much faster acting than a ROM, accessing the BIOS code from RAM results in much faster initialization of the computer.

Shipman, 1:12-22

> The services to be provided by a basic input/output system (BIOS) of a computer system are implemented via a number of independently executable service components. Additionally, the BIOS is provided with a decompression dispatcher for decompressing and dispatching the service components into random access memory (RAM) of the computer system for execution on an as needed basis, and removing the dispatched service components when they are no longer needed. As a result, the service components may be stored in a non-volatile storage in

> a compressed state, allowing more services to be implemented without requiring more non-volatile storage.
>
> More specifically, different basic input/output system (BIOS) functions are split into components, or objects, and selected ones of the components are stored in a compressed state and these compressed components are only decompressed and executed when needed. A decompression dispatcher software takes a request from currently executing BIOS code to decompress another portion of BIOS code that may be needed. The requester can specify whether or not to just decompress the portion of code into memory or to decompress and initialize the decompressed BIOS. The decompression dispatcher can be used to dispatch different portions of BIOS code to different processors in a multi-processor system.

Shipman, 1:35-59

> The BIOS stored in a read only memory (ROM) is organized into a set of components, each being dispatched as an independent executable object after being copied to a shadow memory portion of a random access memory (RAM). Three types of components are defined, initialization components, runtime components and functional components. The initialization components are dispatched during a Power On Self Test (POST) cycle such that the initialization components are active during initialization. The initialization components are terminated prior to loading an operating system. The runtime components are maintained in the uncompressed/decompressed state and executable after loading of the operating system. The functional components are decompressed and dispatched on an as needed basis.

Shipman, 1:60-2:7

> When dispatching components the decompression dispatcher works from a packing list provided in the ROM image by a BIOS build process. To optimize device usage, the binary images that comprise the different components are packed into the final ROM image. The packing list provides a detailed description of the packing as well as other important pieces of information regarding the types of components that have been packed into the ROM.

Shipman, 2:14-22

Appendix A19
Invalidity of U.S. Patent 7,181,608 based on Shipman

Shipman **Claim 1 (Preamble)**

"A method for providing accelerated loading of an operating system, comprising the steps of:"

1.1 maintaining a list of boot data used for booting a computer system;	Shipman, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shipman discloses this limitation:

> The services to be provided by a basic input/output system (BIOS) of a computer system are implemented via a number of independently executable service components. Additionally, the BIOS is provided with a decompression dispatcher for decompressing and dispatching the service components into random access memory (RAM) of the computer system for execution on an as needed basis, and optionally removing the dispatched service components when they are no longer needed. As a result, the service components may be stored in a non-volatile storage in a compressed state, allowing more services to be implemented without requiring more non-volatile storage.

Shipman, Abstract

> When dispatching components the decompression dispatcher works from a packing list provided in the ROM image by a BIOS build process. To optimize device usage, the binary images that comprise the different components are packed into the final ROM image. The packing list provides a detailed description of the packing as well as other important pieces of information regarding the types of components that have been packed into the ROM.

Shipman, 2:14-22

> FIG. 13 is a flow diagram of a BIOS build process that automates compressing and packing components into a final ROM binary image. A list of binary images and associated attributes to include in the final ROM image (1300) and a number of binary image files (1302) are provided to a ROM Packing Utility (1304). The ROM Packing Utility (1304) creates a packed ROM image (1306) and a packing list (1308) that are merged by a merge utility (1310) into a final ROM image

(1312) and a packing map (1314) that are stored in the ROM (10) shown in FIG. 1.

Shipman, 3:55-65

Packing List--The Packing List is generated by the Packing Utility and describes all the attributes of the packed ROM image. This list is used by the decompression dispatcher to dispatch components.

Shipman, 4:33-36

Component Dispatching

Components are dispatched through a utility called the decompression dispatcher. There are two types of dispatching supported by the decompression dispatcher, active and passive.

. . .

When dispatching components the decompression dispatcher works from a Packing List provided in the ROM image by the BIOS build process. To optimize device usage, the binary images that comprise the different components are packed into the final ROM image. The Packing List provides a detailed description of the packing as well as other important pieces of information regarding the types of components that have been packed into the ROM.

Shipman, 5:3-34

Refer to FIG. 3 which is a flow diagram of initializing the decompression dispatcher of FIG. 2. The dispatcher data and executable code is copied (302) from the ROM to the RAM. The dispatcher interrupt vector is installed (304), the memory allocation table is cleared (306) and the component handle table is cleared (308). The compressed BIOS component information in the packing list is fetched from RAM (310). The relocation type field is examined to find what type of relocation is supported, below 1 meg. (312, 314), above 1 meg. (316, 318), or anywhere (320). If the required size is not available (322) allocation error flags are set (324). If the specified load address is not available (326) allocation error flags are set (328).

If the required size is available (322), then the memory allocation table is updated (330). If the end of the component list in the packing list has not been reached (332), then the flow returns to block (310). If the end

of the component list in the packing list has been reached (332), then the flow ends (334).

Shipman, 12:17-35

Refer to FIG. 5 which is a flow diagram of how components are dispatched. The dispatcher assumes only one component from a class is being loaded, and hence the count of components is set to 1. The counter is used as an index into the packing list. The component of a class might contain a number of components referenced by a subclass identifier. A check (504) is made to determine if all components of the class are to be loaded. If yes, then the first subclass identifier is fetched (506) as described in FIG. 7. If the subclass is found in the packing list (508) then the count of the subclass index counter is incremented (510). If the subclass is not found in the packing list (508), then the count of the subclass index counter is not incremented and the flow proceeds to get packing information for the component indexed by the count of the component counter (512). The procedure next decompresses the component at the index count and places the decompressed Code in RAM (514). The details of block (514) are shown in more detail in FIG. 12. At the end of the decompressing procedure, a check (516) is made to determine if there are more components in this class to decompressed. If yes, the component index counter is decremented (518). If no, a check (520) is made to determine if this is the last component actively dispatched or if an option ROM is present. If not, the flow ends (524). If yes, then the return address is adjusted to execute the dispatched component (522).

Shipman, 12:49-13:7

Refer to FIG. 12 which is a flow diagram of the procedure for getting a specified component transferred and stored in RAM in an uncompressed and executable state.

A RAM memory block large enough to hold the specified component is found (1202) and allocated. After the memory space is allocated (1204), the size field in the packing list is accessed to get the size information (1206). The compressed and decompressed size information is obtained from the packing list entry (1206). The Compressed Size field specifies the amount of space (in bytes) consumed by a component when it is in the ROM image. Unitialized Data components use this field to specify the maximum amount of memory required and that they be aligned on a 64K boundary. The Decompressed Size field specifies the amount of space (in bytes) consumed by a component after it has been decompressed and dispatched. As described earlier, components that are

placed in the ROM uncompressed have identical Compressed and Decompressed sizes. Unitialized Data components specify this field to be identical to the Compressed Size field.

If the component is not compressed, then the component is copied (1212) to RAM from the non-volatile storage device used to store the compressed or uncompressed BIOS code. If the component is compressed, then the component is decompressed (1210), stored in an uncompressed state in the shadowed memory portion of RAM, and the procedure stops (1216).

Shipman, 14:15-41

See also Shipman, Figs. 1-13

1.2 initializing a central processing unit of the computer system;	Shipman, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shipman discloses this limitation:

> The services to be provided by a basic input/output system (BIOS) of a computer system are implemented via a number of independently executable service components. Additionally, the BIOS is provided with a decompression dispatcher for decompressing and dispatching the service components into random access memory (RAM) of the computer system for execution on an as needed basis, and optionally removing the dispatched service components when they are no longer needed. As a result, the service components may be stored in a non-volatile storage in a compressed state, allowing more services to be implemented without requiring more non-volatile storage.

Shipman, Abstract

> In IBM compatible personal computers a set of programs called basic input/output system (BIOS) are encoded in read-only memory (ROM). The BIOS facilitates the transfer of data and instructions between a central processing unit (CPU) and peripheral devices such as disk drives. Computer systems are designed to perform functional tests of the BIOS every time the computer is turned on. When the computer is turned on, BIOS is copied to an area of random access memory (RAM) set aside for it. Since a RAM is much faster acting than a ROM, accessing the BIOS code from RAM results in much faster initialization of the computer.

Shipman, 1:12-22

> The BIOS stored in a read only memory (ROM) is organized into a set of components, each being dispatched as an independent executable object after being copied to a shadow memory portion of a random access memory (RAM). Three types of components are defined, initialization components, runtime components and functional

components. The initialization components are dispatched during a Power On Self Test (POST) cycle such that the initialization components are active during initialization. The initialization components are terminated prior to loading an operating system. The runtime components are maintained in the uncompressed/decompressed state and executable after loading of the operating system. The functional components are decompressed and dispatched on an as needed basis.

Shipman, 1:60-2:7

Refer to FIG. 1 which is a block diagram of a computer system in which the present invention is embodied. The computer includes a read only memory, or ROM, (10), a dynamic random access memory, or DRAM, (22), a system memory cache (24), a cache/DRAM controller (20), and central processing units (28, 30). A set of programs called basic input/output system (BIOS) are encoded in ROM (10). The BIOS facilitates the transfer of data and instructions between the central processing units (28, 30) and peripheral devices. In the present invention the BIOS is organized into a set of BIOS components (32 through 54), each of which is capable of being dispatched as an independent executable object. Each BIOS component is a self-contained (link independent) binary image that performs a certain task or function. The BIOS components fall into three major types: initialization components, runtime components and functional components.

Shipman, 2:63-3:12

See also Shipman, Figs. 1-13

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Shipman, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shipman discloses this limitation:

> The services to be provided by a basic input/output system (BIOS) of a computer system are implemented via a number of independently executable service components. Additionally, the BIOS is provided with a decompression dispatcher for decompressing and dispatching the service components into random access memory (RAM) of the computer system for execution on an as needed basis, and optionally removing the dispatched service components when they are no longer needed. As a result, the service components may be stored in a non-volatile storage in a compressed state, allowing more services to be implemented without requiring more non-volatile storage.

Shipman, Abstract

> In IBM compatible personal computers a set of programs called basic input/output system (BIOS) are encoded in read-only memory (ROM). The BIOS facilitates the transfer of data and instructions between a central processing unit (CPU) and peripheral devices such as disk drives. Computer systems are designed to perform functional tests of the BIOS every time the computer is turned on. When the computer is turned on, BIOS is copied to an area of random access memory (RAM) set aside for it. Since a RAM is much faster acting than a ROM, accessing the BIOS code from RAM results in much faster initialization of the computer.

Shipman, 1:12-22

> The services to be provided by a basic input/output system (BIOS) of a computer system are implemented via a number of independently executable service components. Additionally, the BIOS is provided with a decompression dispatcher for decompressing and dispatching the service components into random access memory (RAM) of the computer system for execution on an as needed basis, and removing the dispatched service components when they are no longer needed. As a result, the service components may be stored in a non-volatile storage in a compressed state, allowing more services to be implemented without requiring more non-volatile storage.
>
> More specifically, different basic input/output system (BIOS) functions are split into components, or objects, and selected ones of the components are stored in a compressed state and these compressed components are only decompressed and executed when needed. A decompression dispatcher software takes a request from currently executing BIOS code to decompress another portion of BIOS code that may be needed. The requester can specify whether or not to just decompress the portion of code into memory or to decompress and initialize the decompressed BIOS. The decompression dispatcher can be used to dispatch different portions of BIOS code to different processors in a multi-processor system.

Shipman, 1:35-59

> The BIOS stored in a read only memory (ROM) is organized into a set of components, each being dispatched as an independent executable object after being copied to a shadow memory portion of a random access memory (RAM). Three types of components are defined, initialization components, runtime components and functional components. The initialization components are dispatched during a Power On Self Test (POST) cycle such that the initialization components are active during initialization. The initialization components are terminated prior to loading an operating system. The runtime components are maintained in the uncompressed/decompressed state and executable after loading of the operating system. The functional components are decompressed and dispatched on an as needed basis.

Shipman, 1:60-2:7

> When dispatching components the decompression dispatcher works from a packing list provided in the ROM image by a BIOS build process. To optimize device usage, the binary images that comprise the

different components are packed into the final ROM image. The packing list provides a detailed description of the packing as well as other important pieces of information regarding the types of components that have been packed into the ROM.

Shipman, 2:14-22

Refer to FIG. 1 which is a block diagram of a computer system in which the present invention is embodied. The computer includes a read only memory, or ROM, (10), a dynamic random access memory, or DRAM, (22), a system memory cache (24), a cache/DRAM controller (20), and central processing units (28, 30). A set of programs called basic input/output system (BIOS) are encoded in ROM (10). The BIOS facilitates the transfer of data and instructions between the central processing units (28, 30) and peripheral devices. In the present invention the BIOS is organized into a set of BIOS components (32 through 54), each of which is capable of being dispatched as an independent executable object. Each BIOS component is a self-contained (link independent) binary image that performs a certain task or function. The BIOS components fall into three major types: initialization components, runtime components and functional components.

Shipman, 2:63-3:12

Compressed code is the code that resides in the ROM in a compressed state. Decompressed code is code that has been decompressed from its compressed state inside the ROM and now resides in DRAM (22) shadowed memory, in a decompressed state. Uncompressed code is code that resides inside the ROM in an uncompressed state. This is code that was never compressed.

Shipman, 3:22-28

FIG. 13 is a flow diagram of a BIOS build process that automates compressing and packing components into a final ROM binary image. A list of binary images and associated attributes to include in the final ROM image (1300) and a number of binary image files (1302) are provided to a ROM Packing Utility (1304). The ROM Packing Utility (1304) creates a packed ROM image (1306) and a packing list (1308) that are merged by a merge utility (1310) into a final ROM image (1312) and a packing map (1314) that are stored in the ROM (10) shown in FIG. 1.

Shipman, 3:55-65

> Packing List--The Packing List is generated by the Packing Utility and describes all the attributes of the packed ROM image. This list is used by the decompression dispatcher to dispatch components.

Shipman, 4:33-36

> Component Dispatching
>
> Components are dispatched through a utility called the decompression dispatcher. There are two types of dispatching supported by the decompression dispatcher, active and passive.
>
> . . .
>
> When dispatching components the decompression dispatcher works from a Packing List provided in the ROM image by the BIOS build process. To optimize device usage, the binary images that comprise the different components are packed into the final ROM image. The Packing List provides a detailed description of the packing as well as other important pieces of information regarding the types of components that have been packed into the ROM.

Shipman, 5:3-34

> Refer to FIG. 3 which is a flow diagram of initializing the decompression dispatcher of FIG. 2. The dispatcher data and executable code is copied (302) from the ROM to the RAM. The dispatcher interrupt vector is installed (304), the memory allocation table is cleared (306) and the component handle table is cleared (308). The compressed BIOS component information in the packing list is fetched from RAM (310). The relocation type field is examined to find what type of relocation is supported, below 1 meg. (312, 314), above 1 meg. (316, 318), or anywhere (320). If the required size is not available (322) allocation error flags are set (324). If the specified load address is not available (326) allocation error flags are set (328).
>
> If the required size is available (322), then the memory allocation table is updated (330). If the end of the component list in the packing list has not been reached (332), then the flow returns to block (310). If the end of the component list in the packing list has been reached (332), then the flow ends (334).

Shipman, 12:17-35

> Refer to FIG. 5 which is a flow diagram of how components are dispatched. The dispatcher assumes only one component from a class is being loaded, and hence the count of components is set to 1. The counter is used as an index into the packing list. The component of a class might contain a number of components referenced by a subclass identifier. A check (504) is made to determine if all components of the class are to be loaded. If yes, then the first subclass identifier is fetched (506) as described in FIG. 7. If the subclass is found in the packing list (508) then the count of the subclass index counter is incremented (510). If the subclass is not found in the packing list (508), then the count of the subclass index counter is not incremented and the flow proceeds to get packing information for the component indexed by the count of the component counter (512). The procedure next decompresses the component at the index count and places the decompressed Code in RAM (514). The details of block (514) are shown in more detail in FIG. 12. At the end of the decompressing procedure, a check (516) is made to determine if there are more components in this class to decompressed. If yes, the component index counter is decremented (518). If no, a check (520) is made to determine if this is the last component actively dispatched or if an option ROM is present. If not, the flow ends (524). If yes, then the return address is adjusted to execute the dispatched component (522).

Shipman, 12:49-13:7

> Refer to FIG. 12 which is a flow diagram of the procedure for getting a specified component transferred and stored in RAM in an uncompressed and executable state.
>
> A RAM memory block large enough to hold the specified component is found (1202) and allocated. After the memory space is allocated (1204), the size field in the packing list is accessed to get the size information (1206). The compressed and decompressed size information is obtained from the packing list entry (1206). The Compressed Size field specifies the amount of space (in bytes) consumed by a component when it is in the ROM image. Unitialized Data components use this field to specify the maximum amount of memory required and that they be aligned on a 64K boundary. The Decompressed Size field specifies the amount of space (in bytes) consumed by a component after it has been decompressed and dispatched. As described earlier, components that are placed in the ROM uncompressed have identical Compressed and Decompressed sizes. Unitialized Data components specify this field to be identical to the Compressed Size field.

If the component is not compressed, then the component is copied (1212) to RAM from the non-volatile storage device used to store the compressed or uncompressed BIOS code. If the component is compressed, then the component is decompressed (1210), stored in an uncompressed state in the shadowed memory portion of RAM, and the procedure stops (1216).

Shipman, 14:15-41

See also Shipman, Figs. 1-13

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Shipman, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shipman discloses this limitation:

> The services to be provided by a basic input/output system (BIOS) of a computer system are implemented via a number of independently executable service components. Additionally, the BIOS is provided with a decompression dispatcher for decompressing and dispatching the service components into random access memory (RAM) of the computer system for execution on an as needed basis, and optionally removing the dispatched service components when they are no longer needed. As a result, the service components may be stored in a non-volatile storage in a compressed state, allowing more services to be implemented without requiring more non-volatile storage.

Shipman, Abstract

> In IBM compatible personal computers a set of programs called basic input/output system (BIOS) are encoded in read-only memory (ROM). The BIOS facilitates the transfer of data and instructions between a central processing unit (CPU) and peripheral devices such as disk drives. Computer systems are designed to perform functional tests of the BIOS every time the computer is turned on. When the computer is

> turned on, BIOS is copied to an area of random access memory (RAM) set aside for it. Since a RAM is much faster acting than a ROM, accessing the BIOS code from RAM results in much faster initialization of the computer.

Shipman, 1:12-22

> The services to be provided by a basic input/output system (BIOS) of a computer system are implemented via a number of independently executable service components. Additionally, the BIOS is provided with a decompression dispatcher for decompressing and dispatching the service components into random access memory (RAM) of the computer system for execution on an as needed basis, and removing the dispatched service components when they are no longer needed. As a result, the service components may be stored in a non-volatile storage in a compressed state, allowing more services to be implemented without requiring more non-volatile storage.
>
> More specifically, different basic input/output system (BIOS) functions are split into components, or objects, and selected ones of the components are stored in a compressed state and these compressed components are only decompressed and executed when needed. A decompression dispatcher software takes a request from currently executing BIOS code to decompress another portion of BIOS code that may be needed. The requester can specify whether or not to just decompress the portion of code into memory or to decompress and initialize the decompressed BIOS. The decompression dispatcher can be used to dispatch different portions of BIOS code to different processors in a multi-processor system.

Shipman, 1:35-59

> The BIOS stored in a read only memory (ROM) is organized into a set of components, each being dispatched as an independent executable object after being copied to a shadow memory portion of a random access memory (RAM). Three types of components are defined, initialization components, runtime components and functional components. The initialization components are dispatched during a Power On Self Test (POST) cycle such that the initialization components are active during initialization. The initialization components are terminated prior to loading an operating system. The runtime components are maintained in the uncompressed/decompressed state and executable after loading of the operating system. The

functional components are decompressed and dispatched on an as needed basis.

Shipman, 1:60-2:7

When dispatching components the decompression dispatcher works from a packing list provided in the ROM image by a BIOS build process. To optimize device usage, the binary images that comprise the different components are packed into the final ROM image. The packing list provides a detailed description of the packing as well as other important pieces of information regarding the types of components that have been packed into the ROM.

Shipman, 2:14-22

Refer to FIG. 1 which is a block diagram of a computer system in which the present invention is embodied. The computer includes a read only memory, or ROM, (10), a dynamic random access memory, or DRAM, (22), a system memory cache (24), a cache/DRAM controller (20), and central processing units (28, 30). A set of programs called basic input/output system (BIOS) are encoded in ROM (10). The BIOS facilitates the transfer of data and instructions between the central processing units (28, 30) and peripheral devices. In the present invention the BIOS is organized into a set of BIOS components (32 through 54), each of which is capable of being dispatched as an independent executable object. Each BIOS component is a self-contained (link independent) binary image that performs a certain task or function. The BIOS components fall into three major types: initialization components, runtime components and functional components.

Shipman, 2:63-3:12

Compressed code is the code that resides in the ROM in a compressed state. Decompressed code is code that has been decompressed from its compressed state inside the ROM and now resides in DRAM (22) shadowed memory, in a decompressed state. Uncompressed code is code that resides inside the ROM in an uncompressed state. This is code that was never compressed.

Shipman, 3:22-28

FIG. 13 is a flow diagram of a BIOS build process that automates compressing and packing components into a final ROM binary image.

> A list of binary images and associated attributes to include in the final ROM image (1300) and a number of binary image files (1302) are provided to a ROM Packing Utility (1304). The ROM Packing Utility (1304) creates a packed ROM image (1306) and a packing list (1308) that are merged by a merge utility (1310) into a final ROM image (1312) and a packing map (1314) that are stored in the ROM (10) shown in FIG. 1.

Shipman, 3:55-65

> Packing List--The Packing List is generated by the Packing Utility and describes all the attributes of the packed ROM image. This list is used by the decompression dispatcher to dispatch components.

Shipman, 4:33-36

> Component Dispatching
>
> Components are dispatched through a utility called the decompression dispatcher. There are two types of dispatching supported by the decompression dispatcher, active and passive.
>
> . . .
>
> When dispatching components the decompression dispatcher works from a Packing List provided in the ROM image by the BIOS build process. To optimize device usage, the binary images that comprise the different components are packed into the final ROM image. The Packing List provides a detailed description of the packing as well as other important pieces of information regarding the types of components that have been packed into the ROM.

Shipman, 5:3-34

> Refer to FIG. 3 which is a flow diagram of initializing the decompression dispatcher of FIG. 2. The dispatcher data and executable code is copied (302) from the ROM to the RAM. The dispatcher interrupt vector is installed (304), the memory allocation table is cleared (306) and the component handle table is cleared (308). The compressed BIOS component information in the packing list is fetched from RAM (310). The relocation type field is examined to find what type of relocation is supported, below 1 meg. (312, 314), above 1 meg. (316, 318), or anywhere (320). If the required size is not available (322)

allocation error flags are set (324). If the specified load address is not available (326) allocation error flags are set (328).

If the required size is available (322), then the memory allocation table is updated (330). If the end of the component list in the packing list has not been reached (332), then the flow returns to block (310). If the end of the component list in the packing list has been reached (332), then the flow ends (334).

Shipman, 12:17-35

Refer to FIG. 5 which is a flow diagram of how components are dispatched. The dispatcher assumes only one component from a class is being loaded, and hence the count of components is set to 1. The counter is used as an index into the packing list. The component of a class might contain a number of components referenced by a subclass identifier. A check (504) is made to determine if all components of the class are to be loaded. If yes, then the first subclass identifier is fetched (506) as described in FIG. 7. If the subclass is found in the packing list (508) then the count of the subclass index counter is incremented (510). If the subclass is not found in the packing list (508), then the count of the subclass index counter is not incremented and the flow proceeds to get packing information for the component indexed by the count of the component counter (512). The procedure next decompresses the component at the index count and places the decompressed Code in RAM (514). The details of block (514) are shown in more detail in FIG. 12. At the end of the decompressing procedure, a check (516) is made to determine if there are more components in this class to decompressed. If yes, the component index counter is decremented (518). If no, a check (520) is made to determine if this is the last component actively dispatched or if an option ROM is present. If not, the flow ends (524). If yes, then the return address is adjusted to execute the dispatched component (522).

Shipman, 12:49-13:7

Refer to FIG. 12 which is a flow diagram of the procedure for getting a specified component transferred and stored in RAM in an uncompressed and executable state.

A RAM memory block large enough to hold the specified component is found (1202) and allocated. After the memory space is allocated (1204), the size field in the packing list is accessed to get the size information (1206). The compressed and decompressed size information is obtained

from the packing list entry (1206). The Compressed Size field specifies the amount of space (in bytes) consumed by a component when it is in the ROM image. Unitialized Data components use this field to specify the maximum amount of memory required and that they be aligned on a 64K boundary. The Decompressed Size field specifies the amount of space (in bytes) consumed by a component after it has been decompressed and dispatched. As described earlier, components that are placed in the ROM uncompressed have identical Compressed and Decompressed sizes. Unitialized Data components specify this field to be identical to the Compressed Size field.

If the component is not compressed, then the component is copied (1212) to RAM from the non-volatile storage device used to store the compressed or uncompressed BIOS code. If the component is compressed, then the component is decompressed (1210), stored in an uncompressed state in the shadowed memory portion of RAM, and the procedure stops (1216).

Shipman, 14:15-41

See also Shipman, Figs. 1-13

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Shipman, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shipman discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

> As more new features are designed into computers, more code has to be stored into the BIOS stored on the ROM. The size of the ROM must be increased to accommodate the additional code, resulting in additional manufacturing costs. It is desirable to decreases product cost by reducing the size of a ROM required to store the BIOS and all of its components. This can be done by decompressing the code as it is stored on the ROM and then decompressing the code at the time it is copied to the RAM.

Shipman, 1:23-31

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Shipman, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shipman discloses this limitation:

See Claims 1.3, and 1.4 above.

> Refer to FIG. 1 which is a block diagram of a computer system in which the present invention is embodied. The computer includes a read only memory, or ROM, (10), a dynamic random access memory, or DRAM, (22), a system memory cache (24), a cache/DRAM controller (20), and central processing units (28, 30). A set of programs called basic input/output system (BIOS) are encoded in ROM (10). The BIOS facilitates the transfer of data and instructions between the central processing units (28, 30) and peripheral devices. In the present invention the BIOS is organized into a set of BIOS components (32 through 54), each of which is capable of being dispatched as an independent executable object. Each BIOS component is a self-contained (link independent) binary image that performs a certain task or function. The BIOS components fall into three major types: initialization components, runtime components and functional components.

Shipman, 2:63-3:12

> System random access memory, RAM, (22) and system memory cache (24) am connected to the Cache/DRAM Controller (20) via a local memory bus. The CPU 1 (28) and the CPU 2 (30) is connected to the host to PCI bridge device (18) via the Host CPU Bus (26).

Shipman, 3:50-54

4. The method of claim 1, further comprising updating the list of boot data.	Shipman, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shipman discloses this limitation:

See Claim 1.1 above.

> The services to be provided by a basic input/output system (BIOS) of a computer system are implemented via a number of independently executable service components. Additionally, the BIOS is provided with a decompression dispatcher for decompressing and dispatching the service components into random access memory (RAM) of the computer system for execution on an as needed basis, and optionally removing the dispatched service components when they are no longer needed. As a result, the service components may be stored in a non-volatile storage in a compressed state, allowing more services to be implemented without requiring more non-volatile storage.

Shipman, Abstract

> When dispatching components the decompression dispatcher works from a packing list provided in the ROM image by a BIOS build process. To optimize device usage, the binary images that comprise the different components are packed into the final ROM image. The packing list provides a detailed description of the packing as well as other important pieces of information regarding the types of components that have been packed into the ROM.

Shipman, 2:14-22

> FIG. 13 is a flow diagram of a BIOS build process that automates compressing and packing components into a final ROM binary image. A list of binary images and associated attributes to include in the final ROM image (1300) and a number of binary image files (1302) are provided to a ROM Packing Utility (1304). The ROM Packing Utility (1304) creates a packed ROM image (1306) and a packing list (1308) that are merged by a merge utility (1310) into a final ROM image

(1312) and a packing map (1314) that are stored in the ROM (10) shown in FIG. 1.

Shipman, 3:55-65

Packing List--The Packing List is generated by the Packing Utility and describes all the attributes of the packed ROM image. This list is used by the decompression dispatcher to dispatch components.

Shipman, 4:33-36

Component Dispatching

Components are dispatched through a utility called the decompression dispatcher. There are two types of dispatching supported by the decompression dispatcher, active and passive.

. . .

When dispatching components the decompression dispatcher works from a Packing List provided in the ROM image by the BIOS build process. To optimize device usage, the binary images that comprise the different components are packed into the final ROM image. The Packing List provides a detailed description of the packing as well as other important pieces of information regarding the types of components that have been packed into the ROM.

Shipman, 5:3-34

Refer to FIG. 3 which is a flow diagram of initializing the decompression dispatcher of FIG. 2. The dispatcher data and executable code is copied (302) from the ROM to the RAM. The dispatcher interrupt vector is installed (304), the memory allocation table is cleared (306) and the component handle table is cleared (308). The compressed BIOS component information in the packing list is fetched from RAM (310). The relocation type field is examined to find what type of relocation is supported, below 1 meg. (312, 314), above 1 meg. (316, 318), or anywhere (320). If the required size is not available (322) allocation error flags are set (324). If the specified load address is not available (326) allocation error flags are set (328).

If the required size is available (322), then the memory allocation table is updated (330). If the end of the component list in the packing list has not been reached (332), then the flow returns to block (310). If the end

of the component list in the packing list has been reached (332), then the flow ends (334).

Shipman, 12:17-35

Refer to FIG. 5 which is a flow diagram of how components are dispatched. The dispatcher assumes only one component from a class is being loaded, and hence the count of components is set to 1. The counter is used as an index into the packing list. The component of a class might contain a number of components referenced by a subclass identifier. A check (504) is made to determine if all components of the class are to be loaded. If yes, then the first subclass identifier is fetched (506) as described in FIG. 7. If the subclass is found in the packing list (508) then the count of the subclass index counter is incremented (510). If the subclass is not found in the packing list (508), then the count of the subclass index counter is not incremented and the flow proceeds to get packing information for the component indexed by the count of the component counter (512). The procedure next decompresses the component at the index count and places the decompressed Code in RAM (514). The details of block (514) are shown in more detail in FIG. 12. At the end of the decompressing procedure, a check (516) is made to determine if there are more components in this class to decompressed. If yes, the component index counter is decremented (518). If no, a check (520) is made to determine if this is the last component actively dispatched or if an option ROM is present. If not, the flow ends (524). If yes, then the return address is adjusted to execute the dispatched component (522).

Shipman, 12:49-13:7

Refer to FIG. 12 which is a flow diagram of the procedure for getting a specified component transferred and stored in RAM in an uncompressed and executable state.

A RAM memory block large enough to hold the specified component is found (1202) and allocated. After the memory space is allocated (1204), the size field in the packing list is accessed to get the size information (1206). The compressed and decompressed size information is obtained from the packing list entry (1206). The Compressed Size field specifies the amount of space (in bytes) consumed by a component when it is in the ROM image. Unitialized Data components use this field to specify the maximum amount of memory required and that they be aligned on a 64K boundary. The Decompressed Size field specifies the amount of space (in bytes) consumed by a component after it has been decompressed and dispatched. As described earlier, components that are

placed in the ROM uncompressed have identical Compressed and Decompressed sizes. Unitialized Data components specify this field to be identical to the Compressed Size field.

If the component is not compressed, then the component is copied (1212) to RAM from the non-volatile storage device used to store the compressed or uncompressed BIOS code. If the component is compressed, then the component is decompressed (1210), stored in an uncompressed state in the shadowed memory portion of RAM, and the procedure stops (1216).

Shipman, 14:15-41

See also Shipman, Figs. 1-13

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Shipman, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Shipman, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Shipman, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Shipman, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shipman discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

> Refer to FIG. 1 which is a block diagram of a computer system in which the present invention is embodied. The computer includes a read only memory, or ROM, (10), a dynamic random access memory, or DRAM, (22), a system memory cache (24), a cache/DRAM controller (20), and central processing units (28, 30). A set of programs called basic input/output system (BIOS) are encoded in ROM (10). The BIOS facilitates the transfer of data and instructions between the central processing units (28, 30) and peripheral devices. In the present invention the BIOS is organized into a set of BIOS components (32 through 54), each of which is capable of being dispatched as an independent executable object. Each BIOS component is a self-contained (link independent) binary image that performs a certain task or function. The BIOS components fall into three major types: initialization components, runtime components and functional components.

Shipman, 2:63-3:12

> System random access memory, RAM, (22) and system memory cache (24) am connected to the Cache/DRAM Controller (20) via a local memory bus. The CPU 1 (28) and the CPU 2 (30) is connected to the host to PCI bridge device (18) via the Host CPU Bus (26).

Shipman, 3:50-54

7.2 a cache memory device; and;	Shipman, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Shipman, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Shipman, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Shipman, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Shipman, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shipman discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

> As more new features are designed into computers, more code has to be stored into the BIOS stored on the ROM. The size of the ROM must be increased to accommodate the additional code, resulting in additional manufacturing costs. It is desirable to decreases product cost by reducing the size of a ROM required to store the BIOS and all of its components. This can be done by decompressing the code as it is stored on the ROM and then decompressing the code at the time it is copied to the RAM.

Shipman 1:23-32

9.1 maintaining a list of application data associated with an application program;	Shipman, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Shipman, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Shipman, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shipman discloses this limitation:

See Claims 1.4, 2, and 8 above.

> As more new features are designed into computers, more code has to be stored into the BIOS stored on the ROM. The size of the ROM must be increased to accommodate the additional code, resulting in additional manufacturing costs. It is desirable to decreases product cost by reducing the size of a ROM required to store the BIOS and all of its components. This can be done by decompressing the code as it is stored on the ROM and then decompressing the code at the time it is copied to the RAM.

Shipman 1:23-32

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Shipman, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shipman discloses this limitation:

See ***Copy from 936 element 1.1***

> The services to be provided by a basic input/output system (BIOS) of a computer system are implemented via a number of independently executable service components. Additionally, the BIOS is provided with a decompression dispatcher for decompressing and dispatching the service components into random access memory (RAM) of the computer system for execution on an as needed basis, and optionally removing the dispatched service components when they are no longer needed. As a result, the service components may be stored in a non-volatile storage in a compressed state, allowing more services to be implemented without requiring more non-volatile storage.

Shipman, Abstract

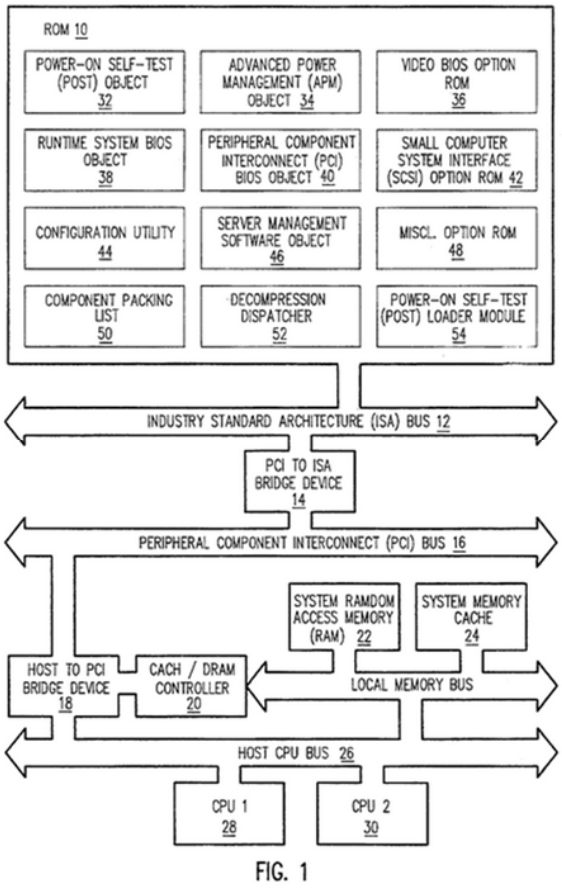


FIG. 1

Shipman, Fig. 1

> In IBM compatible personal computers a set of programs called basic input/output system (BIOS) are encoded in read-only memory (ROM). The BIOS facilitates the transfer of data and instructions between a central processing unit (CPU) and peripheral devices such as disk drives. Computer systems are designed to perform functional tests of the BIOS every time the computer is turned on. When the computer is turned on, BIOS is copied to an area of random access memory (RAM) set aside for it. Since a RAM is much faster acting than a ROM, accessing the BIOS code from RAM results in much faster initialization of the computer.

Shipman, 1:12-22

> The services to be provided by a basic input/output system (BIOS) of a computer system are implemented via a number of independently executable service components. Additionally, the BIOS is provided with a decompression dispatcher for decompressing and dispatching the

> service components into random access memory (RAM) of the computer system for execution on an as needed basis, and removing the dispatched service components when they are no longer needed. As a result, the service components may be stored in a non-volatile storage in a compressed state, allowing more services to be implemented without requiring more non-volatile storage.
>
> More specifically, different basic input/output system (BIOS) functions are split into components, or objects, and selected ones of the components are stored in a compressed state and these compressed components are only decompressed and executed when needed. A decompression dispatcher software takes a request from currently executing BIOS code to decompress another portion of BIOS code that may be needed. The requester can specify whether or not to just decompress the portion of code into memory or to decompress and initialize the decompressed BIOS. The decompression dispatcher can be used to dispatch different portions of BIOS code to different processors in a multi-processor system.

Shipman, 1:35-59

> The BIOS stored in a read only memory (ROM) is organized into a set of components, each being dispatched as an independent executable object after being copied to a shadow memory portion of a random access memory (RAM). Three types of components are defined, initialization components, runtime components and functional components. The initialization components are dispatched during a Power On Self Test (POST) cycle such that the initialization components are active during initialization. The initialization components are terminated prior to loading an operating system. The runtime components are maintained in the uncompressed/decompressed state and executable after loading of the operating system. The functional components are decompressed and dispatched on an as needed basis.

Shipman, 1:60-2:7

> When dispatching components the decompression dispatcher works from a packing list provided in the ROM image by a BIOS build process. To optimize device usage, the binary images that comprise the different components are packed into the final ROM image. The packing list provides a detailed description of the packing as well as other important pieces of information regarding the types of components that have been packed into the ROM.

Shipman, 2:14-22

> Refer to FIG. 1 which is a block diagram of a computer system in which the present invention is embodied. The computer includes a read only memory, or ROM, (10), a dynamic random access memory, or DRAM, (22), a system memory cache (24), a cache/DRAM controller (20), and central processing units (28, 30). A set of programs called basic input/output system (BIOS) are encoded in ROM (10). The BIOS facilitates the transfer of data and instructions between the central processing units (28, 30) and peripheral devices. In the present invention the BIOS is organized into a set of BIOS components (32 through 54), each of which is capable of being dispatched as an independent executable object. Each BIOS component is a self-contained (link independent) binary image that performs a certain task or function. The BIOS components fall into three major types: initialization components, runtime components and functional components.

Shipman, 2:63-3:12

> Compressed code is the code that resides in the ROM in a compressed state. Decompressed code is code that has been decompressed from its compressed state inside the ROM and now resides in DRAM (22) shadowed memory, in a decompressed state. Uncompressed code is code that resides inside the ROM in an uncompressed state. This is code that was never compressed.

Shipman, 3:22-28

> FIG. 13 is a flow diagram of a BIOS build process that automates compressing and packing components into a final ROM binary image. A list of binary images and associated attributes to include in the final ROM image (1300) and a number of binary image files (1302) are provided to a ROM Packing Utility (1304). The ROM Packing Utility (1304) creates a packed ROM image (1306) and a packing list (1308) that are merged by a merge utility (1310) into a final ROM image (1312) and a packing map (1314) that are stored in the ROM (10) shown in FIG. 1.

Shipman, 3:55-65

> Packing List--The Packing List is generated by the Packing Utility and describes all the attributes of the packed ROM image. This list is used by the decompression dispatcher to dispatch components.

Shipman, 4:33-36

> Component Dispatching
>
> Components are dispatched through a utility called the decompression dispatcher. There are two types of dispatching supported by the decompression dispatcher, active and passive.
>
> . . .
>
> When dispatching components the decompression dispatcher works from a Packing List provided in the ROM image by the BIOS build process. To optimize device usage, the binary images that comprise the different components are packed into the final ROM image. The Packing List provides a detailed description of the packing as well as other important pieces of information regarding the types of components that have been packed into the ROM.

Shipman, 5:3-34

> Refer to FIG. 3 which is a flow diagram of initializing the decompression dispatcher of FIG. 2. The dispatcher data and executable code is copied (302) from the ROM to the RAM. The dispatcher interrupt vector is installed (304), the memory allocation table is cleared (306) and the component handle table is cleared (308). The compressed BIOS component information in the packing list is fetched from RAM (310). The relocation type field is examined to find what type of relocation is supported, below 1 meg. (312, 314), above 1 meg. (316, 318), or anywhere (320). If the required size is not available (322) allocation error flags are set (324). If the specified load address is not available (326) allocation error flags are set (328).
>
> If the required size is available (322), then the memory allocation table is updated (330). If the end of the component list in the packing list has not been reached (332), then the flow returns to block (310). If the end of the component list in the packing list has been reached (332), then the flow ends (334).

Shipman, 12:17-35

> Refer to FIG. 5 which is a flow diagram of how components are dispatched. The dispatcher assumes only one component from a class is being loaded, and hence the count of components is set to 1. The counter is used as an index into the packing list. The component of a class might contain a number of components referenced by a subclass

identifier. A check (504) is made to determine if all components of the class are to be loaded. If yes, then the first subclass identifier is fetched (506) as described in FIG. 7. If the subclass is found in the packing list (508) then the count of the subclass index counter is incremented (510). If the subclass is not found in the packing list (508), then the count of the subclass index counter is not incremented and the flow proceeds to get packing information for the component indexed by the count of the component counter (512). The procedure next decompresses the component at the index count and places the decompressed Code in RAM (514). The details of block (514) are shown in more detail in FIG. 12. At the end of the decompressing procedure, a check (516) is made to determine if there are more components in this class to decompressed. If yes, the component index counter is decremented (518). If no, a check (520) is made to determine if this is the last component actively dispatched or if an option ROM is present. If not, the flow ends (524). If yes, then the return address is adjusted to execute the dispatched component (522).

Shipman, 12:49-13:7

Refer to FIG. 12 which is a flow diagram of the procedure for getting a specified component transferred and stored in RAM in an uncompressed and executable state.

A RAM memory block large enough to hold the specified component is found (1202) and allocated. After the memory space is allocated (1204), the size field in the packing list is accessed to get the size information (1206). The compressed and decompressed size information is obtained from the packing list entry (1206). The Compressed Size field specifies the amount of space (in bytes) consumed by a component when it is in the ROM image. Unitialized Data components use this field to specify the maximum amount of memory required and that they be aligned on a 64K boundary. The Decompressed Size field specifies the amount of space (in bytes) consumed by a component after it has been decompressed and dispatched. As described earlier, components that are placed in the ROM uncompressed have identical Compressed and Decompressed sizes. Unitialized Data components specify this field to be identical to the Compressed Size field.

If the component is not compressed, then the component is copied (1212) to RAM from the non-volatile storage device used to store the compressed or uncompressed BIOS code. If the component is compressed, then the component is decompressed (1210), stored in an uncompressed state in the shadowed memory portion of RAM, and the procedure stops (1216).

Appendix A19
Invalidity of U.S. Patent 7,181,608 based on Shipman

Shipman, 14:15-41 *See also* Shipman, Figs. 2-13

Shipman Claim 10

"The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device"

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Shipman, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shipman discloses this limitation:

> The services to be provided by a basic input/output system (BIOS) of a computer system are implemented via a number of independently executable service components. Additionally, the BIOS is provided with a decompression dispatcher for decompressing and dispatching the service components into random access memory (RAM) of the computer system for execution on an as needed basis, and optionally removing the dispatched service components when they are no longer needed. As a result, the service components may be stored in a non-volatile storage in a compressed state, allowing more services to be implemented without requiring more non-volatile storage.

Shipman, Abstract

> In IBM compatible personal computers a set of programs called basic input/output system (BIOS) are encoded in read-only memory (ROM). The BIOS facilitates the transfer of data and instructions between a central processing unit (CPU) and peripheral devices such as disk drives. Computer systems are designed to perform functional tests of the BIOS every time the computer is turned on. When the computer is turned on, BIOS is copied to an area of random access memory (RAM) set aside for it. Since a RAM is much faster acting than a ROM, accessing the BIOS code from RAM results in much faster initialization of the computer.

Shipman, 1:12-22

> The services to be provided by a basic input/output system (BIOS) of a computer system are implemented via a number of independently executable service components. Additionally, the BIOS is provided with a decompression dispatcher for decompressing and dispatching the service components into random access memory (RAM) of the computer system for execution on an as needed basis, and removing the

> dispatched service components when they are no longer needed. As a result, the service components may be stored in a non-volatile storage in a compressed state, allowing more services to be implemented without requiring more non-volatile storage.
>
> More specifically, different basic input/output system (BIOS) functions are split into components, or objects, and selected ones of the components are stored in a compressed state and these compressed components are only decompressed and executed when needed. A decompression dispatcher software takes a request from currently executing BIOS code to decompress another portion of BIOS code that may be needed. The requester can specify whether or not to just decompress the portion of code into memory or to decompress and initialize the decompressed BIOS. The decompression dispatcher can be used to dispatch different portions of BIOS code to different processors in a multi-processor system.

Shipman, 1:35-59

> The BIOS stored in a read only memory (ROM) is organized into a set of components, each being dispatched as an independent executable object after being copied to a shadow memory portion of a random access memory (RAM). Three types of components are defined, initialization components, runtime components and functional components. The initialization components are dispatched during a Power On Self Test (POST) cycle such that the initialization components are active during initialization. The initialization components are terminated prior to loading an operating system. The runtime components are maintained in the uncompressed/decompressed state and executable after loading of the operating system. The functional components are decompressed and dispatched on an as needed basis.

Shipman, 1:60-2:7

> When dispatching components the decompression dispatcher works from a packing list provided in the ROM image by a BIOS build process. To optimize device usage, the binary images that comprise the different components are packed into the final ROM image. The packing list provides a detailed description of the packing as well as other important pieces of information regarding the types of components that have been packed into the ROM.

Shipman, 2:14-22

> Refer to FIG. 1 which is a block diagram of a computer system in which the present invention is embodied. The computer includes a read only memory, or ROM, (10), a dynamic random access memory, or DRAM, (22), a system memory cache (24), a cache/DRAM controller (20), and central processing units (28, 30). A set of programs called basic input/output system (BIOS) are encoded in ROM (10). The BIOS facilitates the transfer of data and instructions between the central processing units (28, 30) and peripheral devices. In the present invention the BIOS is organized into a set of BIOS components (32 through 54), each of which is capable of being dispatched as an independent executable object. Each BIOS component is a self-contained (link independent) binary image that performs a certain task or function. The BIOS components fall into three major types: initialization components, runtime components and functional components.

Shipman, 2:63-3:12

> Compressed code is the code that resides in the ROM in a compressed state. Decompressed code is code that has been decompressed from its compressed state inside the ROM and now resides in DRAM (22) shadowed memory, in a decompressed state. Uncompressed code is code that resides inside the ROM in an uncompressed state. This is code that was never compressed.

Shipman, 3:22-28

> FIG. 13 is a flow diagram of a BIOS build process that automates compressing and packing components into a final ROM binary image. A list of binary images and associated attributes to include in the final ROM image (1300) and a number of binary image files (1302) are provided to a ROM Packing Utility (1304). The ROM Packing Utility (1304) creates a packed ROM image (1306) and a packing list (1308) that are merged by a merge utility (1310) into a final ROM image (1312) and a packing map (1314) that are stored in the ROM (10) shown in FIG. 1.

Shipman, 3:55-65

> Packing List--The Packing List is generated by the Packing Utility and describes all the attributes of the packed ROM image. This list is used by the decompression dispatcher to dispatch components.

Shipman, 4:33-36

> Component Dispatching
>
> Components are dispatched through a utility called the decompression dispatcher. There are two types of dispatching supported by the decompression dispatcher, active and passive.
>
> . . .
>
> When dispatching components the decompression dispatcher works from a Packing List provided in the ROM image by the BIOS build process. To optimize device usage, the binary images that comprise the different components are packed into the final ROM image. The Packing List provides a detailed description of the packing as well as other important pieces of information regarding the types of components that have been packed into the ROM.

Shipman, 5:3-34

> Refer to FIG. 3 which is a flow diagram of initializing the decompression dispatcher of FIG. 2. The dispatcher data and executable code is copied (302) from the ROM to the RAM. The dispatcher interrupt vector is installed (304), the memory allocation table is cleared (306) and the component handle table is cleared (308). The compressed BIOS component information in the packing list is fetched from RAM (310). The relocation type field is examined to find what type of relocation is supported, below 1 meg. (312, 314), above 1 meg. (316, 318), or anywhere (320). If the required size is not available (322) allocation error flags are set (324). If the specified load address is not available (326) allocation error flags are set (328).
>
> If the required size is available (322), then the memory allocation table is updated (330). If the end of the component list in the packing list has not been reached (332), then the flow returns to block (310). If the end of the component list in the packing list has been reached (332), then the flow ends (334).

Shipman, 12:17-35

> Refer to FIG. 5 which is a flow diagram of how components are dispatched. The dispatcher assumes only one component from a class is being loaded, and hence the count of components is set to 1. The counter is used as an index into the packing list. The component of a class might contain a number of components referenced by a subclass identifier. A check (504) is made to determine if all components of the class are to be loaded. If yes, then the first subclass identifier is fetched

(506) as described in FIG. 7. If the subclass is found in the packing list (508) then the count of the subclass index counter is incremented (510). If the subclass is not found in the packing list (508), then the count of the subclass index counter is not incremented and the flow proceeds to get packing information for the component indexed by the count of the component counter (512). The procedure next decompresses the component at the index count and places the decompressed Code in RAM (514). The details of block (514) are shown in more detail in FIG. 12. At the end of the decompressing procedure, a check (516) is made to determine if there are more components in this class to decompressed. If yes, the component index counter is decremented (518). If no, a check (520) is made to determine if this is the last component actively dispatched or if an option ROM is present. If not, the flow ends (524). If yes, then the return address is adjusted to execute the dispatched component (522).

Shipman, 12:49-13:7

Refer to FIG. 12 which is a flow diagram of the procedure for getting a specified component transferred and stored in RAM in an uncompressed and executable state.

A RAM memory block large enough to hold the specified component is found (1202) and allocated. After the memory space is allocated (1204), the size field in the packing list is accessed to get the size information (1206). The compressed and decompressed size information is obtained from the packing list entry (1206). The Compressed Size field specifies the amount of space (in bytes) consumed by a component when it is in the ROM image. Unitialized Data components use this field to specify the maximum amount of memory required and that they be aligned on a 64K boundary. The Decompressed Size field specifies the amount of space (in bytes) consumed by a component after it has been decompressed and dispatched. As described earlier, components that are placed in the ROM uncompressed have identical Compressed and Decompressed sizes. Unitialized Data components specify this field to be identical to the Compressed Size field.

If the component is not compressed, then the component is copied (1212) to RAM from the non-volatile storage device used to store the compressed or uncompressed BIOS code. If the component is compressed, then the component is decompressed (1210), stored in an uncompressed state in the shadowed memory portion of RAM, and the procedure stops (1216).

Shipman, 14:15-41

See also Shipman, Figs. 1-13

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Shipman, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Shipman, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Shipman discloses this limitation:

See Claims 1.3 and 1.4 above.

In IBM compatible personal computers a set of programs called basic input/output system (BIOS) are encoded in read-only memory (ROM). The BIOS facilitates the transfer of data and instructions between a central processing unit (CPU) and peripheral devices such as disk drives. Computer systems are designed to perform functional tests of the BIOS every time the computer is turned on. When the computer is turned on, BIOS is copied to an area of random access memory (RAM) set aside for it. Since a RAM is much faster acting than a ROM, accessing the BIOS code from RAM results in much faster initialization of the computer.

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Shipman, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Shipman, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 1.3, 1.4, and 15 above.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Shipman, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Shipman, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Shipman, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Shipman, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Shipman, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Shipman, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Shipman, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Shipman, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Shipman, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Shipman, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Shipman, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Shipman, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Shipman, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Shipman, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Shipman, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Shipman, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Shipman, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Shipman, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 15 and 27 above	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Shipman, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Shipman discloses this limitation: *See* Claims 16 and 27 above	

Appendix A20
Invalidity of U.S. Patent 7,181,608 based on Sukegawa

U.S. Patent No. 5,860,083 to Sukegawa ("Sukegawa") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

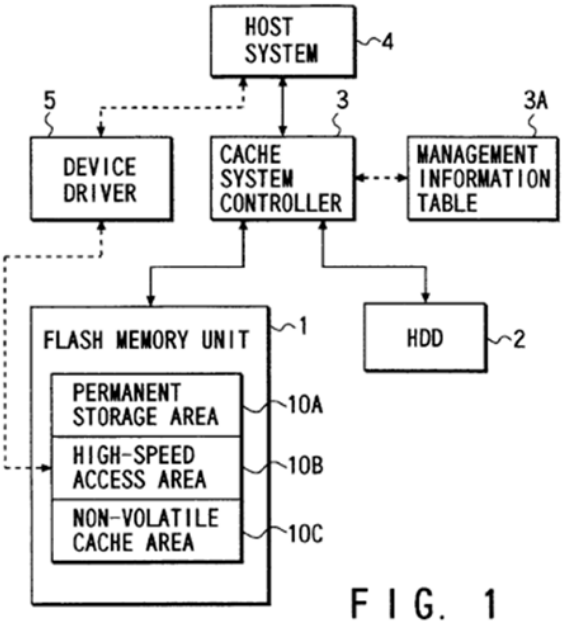
In addition, Apple incorporates by reference, as if set forth fully herein, all arguments related to Sukegawa in pending inter partes review petitions IPR2016-1365, IPR2016-01366, IPR2016-01737, and IPR2016-01738.

Appendix A20
Invalidity of U.S. Patent 7,181,608 based on Sukegawa

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Sukegawa, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Sukegawa discloses this claim limitation:



F I G. 1

Sukegawa, Fig. 1. *See also* Sukegawa, 5:14-6:17, 6:19-58, 7:28-55.

1.1 maintaining a list of boot data used for booting a computer system;	Sukegawa, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Sukegawa discloses this claim limitation:

> "The host system 4 refers to a computer body comprising a CPU of the computer system, a main memory storing the OS and an application program (AP), and other various structural elements. The controller 3 is provided between the host system 4 and the storage units 1 and 2. The controller 3 controls the flash memory unit 1 and HDD 2, as an integrated storage system, in accordance with access requests (read/write commands) issued from the host system 4 to the HDD."

Sukegawa, 4:22-30.

> "According to this system, for example, control information necessary for starting an application program (AP) and an OS, which are frequently used, is stored in the first storage area. Thus, the storage area of the flash memory can be effectively used in accordance with the function and the condition of use of data."

Sukegawa, 2:65-3:3;

> "(First Modification of the First Embodiment)
> FIG. 4 is a flow chart illustrating a first modification of the first embodiment. This modification relates to a system having a mode (data storage mode) for storing the control information necessary for starting the OS in the permanent storage area 10A of flash memory unit 1, for example, when the OS of the host system 4 is started in a series of operations from the turn-on of power to the completion of the starting operation."

Sukegawa, 6:18-26.

> "According to the data storage utility program, the control information, which is pre-stored in the HDD 2 and necessary for starting the OS, is

> read out and stored in the permanent storage area 10A (steps S16 and S17). The controller 3 transfers to the host system 4 the control information necessary for starting the OS read out from the HDD. Based on the control information, the host system 4 starts the OS."

Sukegawa, 6:35-42.

> "According to this system, when the OS is automatically started by the control information read out from the HDD 2 at the time of turning-on of power, the control information is stored in the permanent storage area 10 used as the cache memory area for the HDD 2. Accordingly, when the OS is started at the time of the next turning-on of power, the control information necessary for starting the OS is read out not from the HDD 2 but from the permanent storage area 10 or cache memory area, and the read-out control information is transferred to the host system 4. Thus, the control information can be accessed from the permanent storage area 10A in the flash memory unit 1 having a higher access speed than the HDD 2. As a result, the OS can be started at higher speed."

Sukegawa, 6:45-58.

see also Sukegawa 5:1-7:2, 7:28-55, 9:1-10, 10:33-52, 11:7-19, Fig. 1, Fig. 4, Fig. 5.

1.2 initializing a central processing unit of the computer system;	Sukegawa, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Sukegawa discloses this claim limitation:

> "The host system 4 refers to a computer body comprising a CPU of the computer system, a main memory storing the OS and an application program (AP), and other various structural elements. The controller 3 is provided between the host system 4 and the storage units 1 and 2. The controller 3 controls the flash memory unit 1 and HDD 2, as an integrated storage system, in accordance with access requests (read/write commands) issued from the host system 4 to the HDD."

Sukegawa, 4:22-30.

> "(First Modification of the First Embodiment)
> FIG. 4 is a flow chart illustrating a first modification of the first embodiment. This modification relates to a system having a mode (data storage mode) for storing the control information necessary for starting the OS in the permanent storage area 10A of flash memory unit 1, for example, when the OS of the host system 4 is started in a series of operations from the turn-on of power to the completion of the starting operation."

Sukegawa, 6:18-26.

> "According to the data storage utility program, the control information, which is pre-stored in the HDD 2 and necessary for starting the OS, is read out and stored in the permanent storage area 10A (steps S16 and S17). The controller 3 transfers to the host system 4 the control information necessary for starting the OS read out from the HDD. Based on the control information, the host system 4 starts the OS."

Sukegawa, 6:35-42.

> "According to this system, when the OS is automatically started by the control information read out from the HDD 2 at the time of turning-on of power, the control information is stored in the permanent storage area 10 used as the cache memory area for the HDD 2. Accordingly, when the OS is started at the time of the next turning-on of power, the control information necessary for starting the OS is read out not from the HDD 2 but from the permanent storage area 10 or cache memory area, and the read-out control information is transferred to the host system 4. Thus, the control information can be accessed from the permanent storage area 10A in the flash memory unit 1 having a higher access speed than the HDD 2. As a result, the OS can be started at higher speed."

Sukegawa, 6:45-58.

See also Sukegawa, 6:19-58.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Sukegawa, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Sukegawa discloses this claim limitation:

> "(Data Storage System)
> It is assumed that the data storage system of the present invention is applied to a computer system such as a personal computer. Thus, as shown in FIG. 1, this data storage system has a flash memory unit 1 constituted by a flash EEPROM, a disk drive or an HDD (Hard Disk Drive) 2, and a controller. The controller includes a cache system controller 3 (hereinafter called "controller") for performing a cache function of using the flash memory unit 1 as a cache memory, and a device driver (software) 5 with no cache function.
>
> The device driver 5 has a function of controlling the flash memory unit 1 under the management of the OS (Operation System) of a host system 4. In this invention, in particular, the device driver 5 performs the access control (to be described later) of a highspeed access area lOB or a specific storage area in the flash memory unit 1. The controller 3 performs data input/output control (including cache operation control) for the flash memory unit 1 and HDD 2 via respective device drivers (i.e. a flash memory driver and a hard disk driver)."

Sukegawa, 4:1-21.

> "The first embodiment relates to a system wherein the permanent storage area 10A of flash memory unit 1 is used as a cache memory area."

Sukegawa, 5:10-12.

> "(First Modification of the First Embodiment)
> FIG. 4 is a flow chart illustrating a first modification of the first embodiment. This modification relates to a system having a mode (data storage mode) for storing the control information necessary for starting the OS in the permanent storage area 10A of flash memory unit 1, for example, when the OS of the host system 4 is started in a series of operations from the turn-on of power to the completion of the starting operation."

Sukegawa, 6:18-26.

> "According to the data storage utility program, the control information, which is pre-stored in the HDD 2 and necessary for starting the OS, is read out and stored in the permanent storage area 10A (steps S16 and S17). The controller 3 transfers to the host system 4 the control information necessary for starting the OS read out from the HDD. Based on the control information, the host system 4 starts the OS."

Sukegawa, 6:35-42.

> "According to this system, when the OS is automatically started by the control information read out from the HDD 2 at the time of turning-on of power, the control information is stored in the permanent storage area 10 used as the cache memory area for the HDD 2. Accordingly, when the OS is started at the time of the next turning-on of power, the control information necessary for starting the OS is read out not from the HDD 2 but from the permanent storage area 10 or cache memory area, and the read-out control information is transferred to the host system 4. Thus, the control information can be accessed from the permanent storage area 10A in the flash memory unit 1 having a higher access speed than the HDD 2. As a result, the OS can be started at higher speed."

Sukegawa, 6:45-58.

> "In the system of the present invention, when the host system 4 issues the read command to the HDD 2, the controller 3 determines whether the data to be accessed (e.g. AP control information as mentioned above) is stored in the permanent storage area 10A or non-volatile cache area 10C, which is the cache memory area (or whether the cache

memory area is "hit") (steps S20 and S21), as shown in FIGS. 5. If the data to be accessed is "hit", the controller 3 reads the data from the permanent storage area 10A or non-volatile cache area 10 and transfers the read-out data to the host system 4 ("YES" in step S21; step S25). On the other hand, if the cache memory area is not "hit", the controller 3 accesses the HDD 2, reads out the data to be accessed and transfers the read-out data to the host system 4. In this case, as described above, if the data to be accessed is the permanent data designated by the user, the controller 3 stores it in the permanent storage area 10A ("NO" in step S21; steps S22 and 23). If the data to be accessed is not the permanent data designated by the user, the controller 3 stores the data in the non-volatile cache area 10C which is used as an ordinary cache memory area ("NO" in step S22; step S24). Thus, the data read out from the HDD 2 is stored at least once in the permanent storage area 10A and nonvolatile cache area 10C in flash memory unit 1. In particular, it is assumed that the non-volatile cache area 10C is used independently by the controller 3 and not directly used by
the user's intent."

Sukegawa, 7:28-55

see also Sukegawa 5:1-7:2, 9:1-10, 10:33-52, 11:7-19, Fig 1, Fig. 4, Fig. 5.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Sukegawa, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Sukegawa discloses this claim limitation:

> "(Data Storage System)
> It is assumed that the data storage system of the present invention is applied to a computer system such as a personal computer. Thus, as shown in FIG. 1, this data storage system has a flash memory unit 1 constituted by a flash EEPROM, a disk drive or an HDD (Hard Disk Drive) 2, and a controller. The controller includes a cache system controller 3 (hereinafter called "controller") for performing a cache function of using the flash memory unit 1 as a cache memory, and a device driver (software) 5 with no cache function.
>
> The device driver 5 has a function of controlling the flash memory unit 1 under the management of the OS (Operation System) of a host system 4. In this invention, in particular, the device driver 5 performs the access control (to be described later) of a highspeed access area 10B or a specific storage area in the flash memory unit 1. The controller 3 performs data input/output control (including cache operation control) for the flash memory unit 1 and HDD 2 via respective device drivers (i.e. a flash memory driver and a hard disk driver)."

Sukegawa, 4:1-21.

Sukegawa Claim 2

"The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

> "The first embodiment relates to a system wherein the permanent storage area 10A of flash memory unit 1 is used as a cache memory area."

Sukegawa, 5:10-12.

> "(First Modification of the First Embodiment)
> FIG. 4 is a flow chart illustrating a first modification of the first embodiment. This modification relates to a system having a mode (data storage mode) for storing the control information necessary for starting the OS in the permanent storage area 10A of flash memory unit 1, for example, when the OS of the host system 4 is started in a series of operations from the turn-on of power to the completion of the starting operation."

Sukegawa, 6:18-26.

> "According to the data storage utility program, the control information, which is pre-stored in the HDD 2 and necessary for starting the OS, is read out and stored in the permanent storage area 10A (steps S16 and S17). The controller 3 transfers to the host system 4 the control information necessary for starting the OS read out from the HDD. Based on the control information, the host system 4 starts the OS."

Sukegawa, 6:35-42.

> "According to this system, when the OS is automatically started by the control information read out from the HDD 2 at the time of turning-on of power, the control information is stored in the permanent storage area 10 used as the cache memory area for the HDD 2. Accordingly, when the OS is started at the time of the next turning-on of power, the control information necessary for starting the OS is read out not from the HDD 2 but from the permanent storage area 10 or cache memory area, and the read-out control information is transferred to the host system 4. Thus, the control information can be accessed from the permanent storage area 10A in the flash memory unit 1 having a higher access speed than the HDD 2. As a result, the OS can be started at higher speed."

Sukegawa, 6:45-58.

See also Sukegawa 5:1-7:2, 7:28-55, 9:1-10, 10:33-52, 11:7-19, Fig 1, Fig. 4, Fig. 5.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Sukegawa, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Sukegawa discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> The first embodiment relates to a system wherein the permanent storage area lOA of flash memory unit 1 is used as a cache memory area. In this embodiment, it is supposed that the user desires to start a frequently used application program (AP) at high speed at all times.
>
> The user starts a data storage utility program of the cache system controller 3 via a user interface provided in the host system 4 (step S1). The data storage utility program reads specified data from the HDD 2 and stores the read data in a specified storage area in the flash memory unit 1. In this case, it is assumed that the user sets the permanent storage area 10A in the flash memory unit 1 as the data storage area, at the time of instructing the start of the data storage utility program (step S2).
>
> Then, the user instructs the host system 4 to start the AP (step S3). The host system 4, upon receiving the AP start instruction, issues a read command to the controller 3 in order to read control information necessary for the start of the AP from the HDD 2.
>
> The controller 3 controls the HDD 2, reads out the control information necessary for the start of the AP and transfers the read-out control information to the host system 4 (step S4). At this time, according to the started-up data storage utility program, the controller 3 stores the AP control information read out from the HDD 2 in the permanent storage area 10A of flash memory unit 1 (step S5). When the AP has been

prepared to start, the data storage utility program is stopped by the instruction from the user ("YES" in step S6; step S7). Through these operations, the control information necessary for starting the AP is stored in the permanent storage area 10A in the flash memory unit 1.

Sukegawa 5:10-40

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Sukegawa, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Sukegawa discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

> It is assumed that the data storage system of the present invention is applied to a computer system such as a personal computer. Thus, as shown in FIG. 1, this data storage system has a flash memory unit 1 constituted by a flash EEPROM, a disk drive or an HDD (Hard Disk Drive) 2, and a controller. The controller includes a cache system controller 3 (hereinafter called "controller") for performing a cache function of using the flash memory unit 1 as a cache memory, and a device driver (software) 5 with no cache function.
>
> The device driver 5 has a function of controlling the flash memory unit 1 under the management of the OS (Operation System) of a host system 4. In this invention, in particular, the device driver 5 performs the access control (to be described later) of a highspeed access area 10B or a specific storage area in the flash memory unit 1. The controller 3 performs data input/output control (including cache operation control) for the flash memory unit 1 and HDD 2 via respective device drivers (i.e. a flash memory driver and a hard disk driver).

Sukegawa 4:1-21

4. The method of claim 1, further comprising updating the list of boot data.	Sukegawa, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Sukegawa discloses this limitation:

See Claim 1.1 above.

See also

> "The host system 4 refers to a computer body comprising a CPU of the computer system, a main memory storing the OS and an application program (AP), and other various structural elements. The controller 3 is provided between the host system 4 and the storage units 1 and 2. The controller 3 controls the flash memory unit 1 and HDD 2, as an integrated storage system, in accordance with access requests (read/write commands) issued from the host system 4 to the HDD."

Sukegawa, 4:22-30.

> "According to this system, for example, control information necessary for starting an application program (AP) and an OS, which are frequently used, is stored in the first storage area. Thus, the storage area of the flash memory can be effectively used in accordance with the function and the condition of use of data."

Sukegawa, 2:65-3:3;

> "(First Modification of the First Embodiment)
> FIG. 4 is a flow chart illustrating a first modification of the first embodiment. This modification relates to a system having a mode (data storage mode) for storing the control information necessary for starting the OS in the permanent storage area 10A of flash memory unit 1, for example, when the OS of the host system 4 is started in a series of operations from the turn-on of power to the completion of the starting operation."

Sukegawa, 6:18-26.

> "According to the data storage utility program, the control information, which is pre-stored in the HDD 2 and necessary for starting the OS, is read out and stored in the permanent storage area 10A (steps S16 and S17). The controller 3 transfers to the host system 4 the control information necessary for starting the OS read out from the HDD. Based on the control information, the host system 4 starts the OS."

Sukegawa, 6:35-42.

> "According to this system, when the OS is automatically started by the control information read out from the HDD 2 at the time of turning-on of power, the control information is stored in the permanent storage area 10 used as the cache memory area for the HDD 2. Accordingly, when the OS is started at the time of the next turning-on of power, the control information necessary for starting the OS is read out not from the HDD 2 but from the permanent storage area 10 or cache memory area, and the read-out control information is transferred to the host system 4. Thus, the control information can be accessed from the permanent storage area 10A in the flash memory unit 1 having a higher access speed than the HDD 2. As a result, the OS can be started at higher speed."

Sukegawa, 6:45-58.

> "The permanent storage area 10A and non-volatile cache area 10C function as cache memory areas of the HDD 2. Normally, each time the data in the cache memory area is updated, the updated data is written in the HDD 2. In this embodiment, the user can set the mode of each of the areas 10A and 10C, thereby determining whether or not the updated data should be written in the HDD 2 each time the data is updated."

Sukegawa, 9:19-29.

See also Sukegawa 5:1-7:2, 7:28-55, 9:1-10, 9:53-10:52, 11:7-19, Fig 1, Fig. 4, Fig. 5

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Sukegawa, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Sukegawa, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Sukegawa, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Sukegawa, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Sukegawa discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

> It is assumed that the data storage system of the present invention is applied to a computer system such as a personal computer. Thus, as shown in FIG. 1, this data storage system has a flash memory unit 1 constituted by a flash EEPROM, a disk drive or an HDD (Hard Disk Drive) 2, and a controller. The controller includes a cache system controller 3 (hereinafter called "controller") for performing a cache function of using the flash memory unit 1 as a cache memory, and a device driver (software) 5 with no cache function.
>
> The device driver 5 has a function of controlling the flash memory unit 1 under the management of the OS (Operation System) of a host system 4. In this invention, in particular, the device driver 5 performs the access control (to be described later) of a highspeed access area 10B or a specific storage area in the flash memory unit 1. The controller 3 performs data input/output control (including cache operation control) for the flash memory unit 1 and HDD 2 via respective device drivers (i.e. a flash memory driver and a hard disk driver).

Sukegawa 4:1-21

7.2 a cache memory device; and;	Sukegawa, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Sukegawa, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Sukegawa, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Sukegawa, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Sukegawa, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Sukegawa discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

See also

> The first embodiment relates to a system wherein the permanent storage area lOA of flash memory unit 1 is used as a cache memory area. In this embodiment, it is supposed that the user desires to start a frequently used application program (AP) at high speed at all times.
>
> The user starts a data storage utility program of the cache system controller 3 via a user interface provided in the host system 4 (step S1). The data storage utility program reads specified data from the HDD 2 and stores the read data in a specified storage area in the flash memory unit 1. In this case, it is assumed that the user sets the permanent storage area 10A in the flash memory unit 1 as the data storage area, at the time of instructing the start of the data storage utility program (step S2).

> Then, the user instructs the host system 4 to start the AP (step S3). The host system 4, upon receiving the AP start instruction, issues a read command to the controller 3 in order to read control information necessary for the start of the AP from the HDD 2.
>
> The controller 3 controls the HDD 2, reads out the control information necessary for the start of the AP and transfers the read-out control information to the host system 4 (step S4). At this time, according to the started-up data storage utility program, the controller 3 stores the AP control information read out from the HDD 2 in the permanent storage area 10A of flash memory unit 1 (step S5). When the AP has been prepared to start, the data storage utility program is stopped by the instruction from the user ("YES" in step S6; step S7). Through these operations, the control information necessary for starting the AP is stored in the permanent storage area 10A in the flash memory unit 1.

Sukegawa 5:10-40

9.1 maintaining a list of application data associated with an application program;	Sukegawa, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Sukegawa, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Sukegawa, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Sukegawa discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> The first embodiment relates to a system wherein the permanent storage area lOA of flash memory unit 1 is used as a cache memory area. In this embodiment, it is supposed that the user desires to start a frequently used application program (AP) at high speed at all times.
>
> The user starts a data storage utility program of the cache system controller 3 via a user interface provided in the host system 4 (step S1). The data storage utility program reads specified data from the HDD 2 and stores the read data in a specified storage area in the flash memory unit 1. In this case, it is assumed that the user sets the permanent storage area 10A in the flash memory unit 1 as the data storage area, at the time of instructing the start of the data storage utility program (step S2).
>
> Then, the user instructs the host system 4 to start the AP (step S3). The host system 4, upon receiving the AP start instruction, issues a read command to the controller 3 in order to read control information

necessary for the start of the AP from the HDD 2.

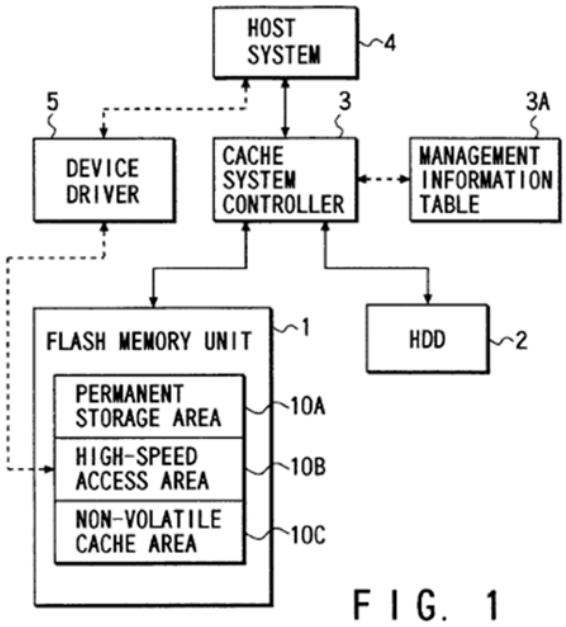
The controller 3 controls the HDD 2, reads out the control information necessary for the start of the AP and transfers the read-out control information to the host system 4 (step S4). At this time, according to the started-up data storage utility program, the controller 3 stores the AP control information read out from the HDD 2 in the permanent storage area 10A of flash memory unit 1 (step S5). When the AP has been prepared to start, the data storage utility program is stopped by the instruction from the user ("YES" in step S6; step S7). Through these operations, the control information necessary for starting the AP is stored in the permanent storage area 10A in the flash memory unit 1.

Sukegawa 5:10-40

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Sukegawa, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Sukegawa discloses this limitation:



F I G. 1

Sukegawa, Fig. 1

> "(Data Storage System)
> It is assumed that the data storage system of the present invention is applied to a computer system such as a personal computer. Thus, as shown in FIG. 1, this data storage system has a flash memory unit 1 constituted by a flash EEPROM, a disk drive or an HDD (Hard Disk Drive) 2, and a controller. The controller includes a cache system

> controller 3 (hereinafter called "controller") for performing a cache function of using the flash memory unit 1 as a cache memory, and a device driver (software) 5 with no cache function.
>
> The device driver 5 has a function of controlling the flash memory unit 1 under the management of the OS (Operation System) of a host system 4. In this invention, in particular, the device driver 5 performs the access control (to be described later) of a highspeed access area lOB or a specific storage area in the flash memory unit 1. The controller 3 performs data input/output control (including cache operation control) for the flash memory unit 1 and HDD 2 via respective device drivers (i.e. a flash memory driver and a hard disk driver).”

Sukegawa, 4:1-21.

> “The host system 4 refers to a computer body comprising a CPU of the computer system, a main memory storing the OS and an application program (AP), and other various structural elements. The controller 3 is provided between the host system 4 and the storage units 1 and 2. The controller 3 controls the flash memory unit 1 and HDD 2, as an integrated storage system, in accordance with access requests (read/write commands) issued from the host system 4 to the HDD.”

Sukegawa, 4:22-30.

> “According to this system, for example, control information necessary for starting an application program (AP) and an OS, which are frequently used, is stored in the first storage area. Thus, the storage area of the flash memory can be effectively used in accordance with the function and the condition of use of data.”

Sukegawa, 2:65-3:3.

> “(First Modification of the First Embodiment)
> FIG. 4 is a flow chart illustrating a first modification of the first embodiment. This modification relates to a system having a mode (data storage mode) for storing the control information necessary for starting the OS in the permanent storage area lOA of flash memory unit 1, for example, when the OS of the host system 4 is started in a series of operations from the turn-on of power to the completion of the starting operation.”

Sukegawa, 6:18-26.

> "According to the data storage utility program, the control information, which is pre-stored in the HDD 2 and necessary for starting the OS, is read out and stored in the permanent storage area 10A (steps S16 and S17). The controller 3 transfers to the host system 4 the control information necessary for starting the OS read out from the HDD. Based on the control information, the host system 4 starts the OS."

Sukegawa, 6:35-42.

> "According to this system, when the OS is automatically started by the control information read out from the HDD 2 at the time of turning-on of power, the control information is stored in the permanent storage area 10 used as the cache memory area for the HDD 2. Accordingly, when the OS is started at the time of the next turning-on of power, the control information necessary for starting the OS is read out not from the HDD 2 but from the permanent storage area 10 or cache memory area, and the read-out control information is transferred to the host system 4. Thus, the control information can be accessed from the permanent storage area 10A in the flash memory unit 1 having a higher access speed than the HDD 2. As a result, the OS can be started at higher speed."

Sukegawa, 6:45-58.

see also Sukegawa 5:1-7:2, 7:28-55, 9:1-10, 10:33-52, 11:7-19, Fig. 4, Fig. 5.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Sukegawa, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Sukegawa discloses this claim limitation:

> "(Data Storage System)
> It is assumed that the data storage system of the present invention is applied to a computer system such as a personal computer. Thus, as shown in FIG. 1, this data storage system has a flash memory unit 1 constituted by a flash EEPROM, a disk drive or an HDD (Hard Disk Drive) 2, and a controller. The controller includes a cache system controller 3 (hereinafter called "controller") for performing a cache function of using the flash memory unit 1 as a cache memory, and a device driver (software) 5 with no cache function.
>
> The device driver 5 has a function of controlling the flash memory unit 1 under the management of the OS (Operation System) of a host system 4. In this invention, in particular, the device driver 5 performs the access control (to be described later) of a highspeed access area lOB or a specific storage area in the flash memory unit 1. The controller 3 performs data input/output control (including cache operation control) for the flash memory unit 1 and HDD 2 via respective device drivers (i.e. a flash memory driver and a hard disk driver)."

Sukegawa, 4:1-21.

> "The first embodiment relates to a system wherein the permanent storage area 10A of flash memory unit 1 is used as a cache memory area."

Sukegawa, 5:10-12.

> "(First Modification of the First Embodiment)
> FIG. 4 is a flow chart illustrating a first modification of the first embodiment. This modification relates to a system having a mode (data storage mode) for storing the control information necessary for starting the OS in the permanent storage area lOA of flash memory unit 1, for

example, when the OS of the host system 4 is started in a series of operations from the turn-on of power to the completion of the starting operation."

Sukegawa, 6:18-26.

"According to the data storage utility program, the control information, which is pre-stored in the HDD 2 and necessary for starting the OS, is read out and stored in the permanent storage area 10A (steps S16 and S17). The controller 3 transfers to the host system 4 the control information necessary for starting the OS read out from the HDD. Based on the control information, the host system 4 starts the OS."

Sukegawa, 6:35-42.

"According to this system, when the OS is automatically started by the control information read out from the HDD 2 at the time of turning-on of power, the control information is stored in the permanent storage area 10 used as the cache memory area for the HDD 2. Accordingly, when the OS is started at the time of the next turning-on of power, the control information necessary for starting the OS is read out not from the HDD 2 but from the permanent storage area 10 or cache memory area, and the read-out control information is transferred to the host system 4. Thus, the control information can be accessed from the permanent storage area 10A in the flash memory unit 1 having a higher access speed than the HDD 2. As a result, the OS can be started at higher speed."

Sukegawa, 6:45-58.

"In the system of the present invention, when the host system 4 issues the read command to the HDD 2, the controller 3 determines whether the data to be accessed (e.g. AP control information as mentioned above) is stored in the permanent storage area 10A or non-volatile cache area 10C, which is the cache memory area (or whether the cache memory area is "hit") (steps S20 and S21), as shown in FIGS. 5. If the data to be accessed is "hit", the controller 3 reads the data from the permanent storage area 10A or non-volatile cache area 10 and transfers the read-out data to the host system 4 ("YES" in step S21; step S25). On the other hand, if the cache memory area is not "hit", the controller 3 accesses the HDD 2, reads out the data to be accessed and transfers the read-out data to the host system 4. In this case, as described above, if the data to be accessed is the permanent data designated by the user, the controller 3 stores it in the permanent storage area 10A ("NO" in step S21; steps S22 and 23). If the data to be accessed is not the permanent data designated by the user, the controller 3 stores the data in

> the non-volatile cache area 10C which is used as an ordinary cache memory area ("NO" in step S22; step S24). Thus, the data read out from the HDD 2 is stored at least once in the permanent storage area 10A and nonvolatile cache area 10C in flash memory unit 1. In particular, it is assumed that the non-volatile cache area 10C is used independently by the controller 3 and not directly used by the user's intent."

Sukegawa, 7:28-55

see also Sukegawa 5:1-7:2, 9:1-10, 10:33-52, 11:7-19, Fig 1, Fig. 4, Fig. 5.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Sukegawa, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Sukegawa, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Sukegawa discloses this limitation:

See Claims 1.3 and 1.4 above.

See also

> "(Data Storage System)
> It is assumed that the data storage system of the present invention is applied to a computer system such as a personal computer. Thus, as shown in FIG. 1, this data storage system has a flash memory unit 1 constituted by a flash EEPROM, a disk drive or an HDD (Hard Disk Drive) 2, and a controller. The controller includes a cache system controller 3 (hereinafter called "controller") for performing a cache function of using the flash memory unit 1 as a cache memory, and a device driver (software) 5 with no cache function.
>
> The device driver 5 has a function of controlling the flash memory unit 1 under the management of the OS (Operation System) of a host system 4. In this invention, in particular, the device driver 5 performs the access control (to be described later) of a highspeed access area lOB or a specific storage area in the flash memory unit 1. The controller 3 performs data input/output control (including cache operation control) for the flash memory unit 1 and HDD 2 via respective device drivers (i.e. a flash memory driver and a hard disk driver)."

Sukegawa, 4:1-21.

> "According to this system, when the OS is automatically started by the control information read out from the HDD 2 at the time of turning-on of power, the control information is stored in the permanent storage area 10 used as the cache memory area for the HDD 2. Accordingly, when the OS is started at the time of the next turning-on of power, the control information necessary for starting the OS is read out not from the HDD 2 but from the permanent storage area 10 or cache memory area, and the

read-out control information is transferred to the host system 4. Thus, the control information can be accessed from the permanent storage area 10A in the flash memory unit 1 having a higher access speed than the HDD 2. As a result, the OS can be started at higher speed."

Sukegawa, 6:45-58.

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Sukegawa, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Sukegawa, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 1.3, 1.4, and 15 above.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Sukegawa, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Sukegawa, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Sukegawa, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Sukegawa, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Sukegawa, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Sukegawa, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Sukegawa, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Sukegawa, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Sukegawa, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Sukegawa, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Sukegawa, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Sukegawa, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Sukegawa, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Sukegawa, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Sukegawa, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Sukegawa, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Sukegawa, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Sukegawa, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 15 and 27 above	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Sukegawa, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Sukegawa discloses this limitation: *See* Claims 16 and 27 above	

Appendix A21
Invalidity of U.S. Patent 7,181,608 based on Surine

U.S. Patent No. 6,212,632 to Surine ("Surine") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A21
Invalidity of U.S. Patent 7,181,608 based on Surine

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Surine, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Surine discloses this limitation:

> "At power-up, boot code stored in the non-volatile memory is executed and begins instantiating the initial operating environment of the embedded computer system. A function pointer table is instantiated in the volatile memory, wherein the function pointer table includes a plurality of entries for a corresponding plurality of instantiated functions, wherein at least one entry is for operating system code stored in the non-volatile memory."

Surine, Abstract.

> "At boot time, or power-up, boot code 202 executes, decompresses compressed code 201 into camera system code 203 and loads camera system code 203 into RAM 102."

Surine, 2:4-7, Fig. 2.

1.1 maintaining a list of boot data used for booting a computer system;	Surine, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Surine discloses this limitation:

> "At power-up, boot code stored in the non-volatile memory is executed and begins instantiating the initial operating environment of the embedded computer system. A function pointer table is instantiated in the volatile memory, wherein the function pointer table includes a plurality of entries for a corresponding plurality of instantiated functions, wherein at least one entry is for operating system code stored in the non-volatile memory."

Surine, Abstract.

> "At power-up, boot code stored in the ROM is executed and begins instantiating the initial operating environment of the embedded system. A function pointer table is instantiated in the RAM. The function pointer table has entries, or function pointers, for each instantiated function such that they can each call each other and pass execution. The function pointer table has entries for functions which are instantiated in ROM and entries for functions which are instantiated in RAM."

Surine, 2:66-3:7.

> "Referring now to FIG. 4, a memory diagram depicting the contents of ROM 310 and RAM 315 is shown. As shown in FIG. 4, ROM 310 stores software including boot code 401, compressed high-use functions 402, and operating system code 403."

Surine, 5:15-19.

> "Function pointer table 510 includes a plurality of entries, or function pointers, which, when called by processing unit 305, redirect program execution to the memory address of a selected routine. The function pointers of function pointer table allow instantiated functions, whether

executing from ROM 310 or RAM 315, to call one another. Initially, the function pointers are initialized to addresses in ROM 310, but after copying to RAM 315, the high-use function pointers are updated to addresses in RAM 315."

Surine, 6:24-29. *See also* 8:54-56.

1.2 initializing a central processing unit of the computer system;	Surine, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Surine discloses this limitation:

> "At power-up, boot code stored in the non-volatile memory is executed and begins instantiating the initial operating environment of the embedded computer system. A function pointer table is instantiated in the volatile memory, wherein the function pointer table includes a plurality of entries for a corresponding plurality of instantiated functions, wherein at least one entry is for operating system code stored in the non-volatile memory."

Surine, Abstract.

> "At power-up, boot code stored in the ROM is executed and begins instantiating the initial operating environment of the embedded system. A function pointer table is instantiated in the RAM. The function pointer table has entries, or function pointers, for each instantiated function such that they can each call each other and pass execution. The function pointer table has entries for functions which are instantiated in ROM and entries for functions which are instantiated in RAM."

Surine, 2:66-3:7.

> "Referring now to FIG. 4, a memory diagram depicting the contents of ROM 310 and RAM 315 is shown. As shown in FIG. 4, ROM 310 stores software including boot code 401, compressed high-use functions 402, and operating system code 403."

Surine, 5:15-19.

> "Function pointer table 510 includes a plurality of entries, or function pointers, which, when called by processing unit 305, redirect program execution to the memory address of a selected routine. The function pointers of function pointer table allow instantiated functions, whether

executing from ROM 310 or RAM 315, to call one another. Initially, the function pointers are initialized to addresses in ROM 310, but after copying to RAM 315, the high-use function pointers are updated to addresses in RAM 315."

Surine, 6:24-29. *See also* 8:54-56.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Surine, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Surine discloses this limitation:

> "Typically, as shown in FIG. 2, the camera system code 203 (e.g., operating system software and its associated data structures, resources, etc.) is stored as compressed code 201 in non-volatile ROM 104."

Surine, 2:1-4, Fig. 2.

> "At boot time, or power-up, boot code 202 executes, decompresses compressed code 201 into camera system code 203 and loads camera system code 203 into RAM 102."

Surine, 2:4-7, Fig. 2.

> "Consequently, these digital cameras and other performance-oriented types of embedded system consumer electronic devices transfer a compressed image of their system code from a non-volatile ROM to a faster RAM at power up. The system code then executes from RAM."

Surine, 2:21-25.

> "In accordance with the present invention, a set of high-use functions are decompressed out of ROM and instantiated in RAM using a patch manager."

Surine, 3:7-9.

> "Referring now to FIG. 4, a memory diagram depicting the contents of ROM 310 and RAM 315 is shown. As shown in FIG. 4, ROM 310 stores software including boot code 401, compressed high-use functions 402, and operating system code 403."

Surine, 5:15-19.

> "Patch manager 405 subsequently executes. Patch manager 405 includes decompression software which decompresses compressed high-use functions 402 and, as described in greater detail in the discussion of FIG. 5 below, loads the resulting decompressed high-use functions 416 into RAM 315."

Surine, 5:35-40.

> "Patch manager 405 then loads the decompressed high-use functions 416 into RAM 315 and updates function pointer table 510 with entries for high-use functions 416."

Surine: 6:32-34.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Surine, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Surine discloses this limitation:

> "At least one high-use function is decompressed out of the non-volatile memory and instantiated in volatile memory. The function pointer table is updated using a patch manager to incorporate an entry for the high-use function(s). The operating system code is executed from the non-volatile memory while the high-use function is executed from the volatile memory."

Surine, Abstract.

> "At boot time, or power-up, boot code 202 executes, decompresses compressed code 201 into camera system code 203 and loads camera system code 203 into RAM 102."

Surine, 2:4-7, Fig. 2.

> "Consequently, these digital cameras and other performance-oriented types of embedded system consumer electronic devices transfer a compressed image of their system code from a non-volatile ROM to a

faster RAM at power up. The system code then executes from RAM."

Surine, 2:21-25.

> "Additionally, the system of the present invention maintains the speed and responsiveness of the device while reducing the amount of RAM used in the device. In comparison to prior art embedded system devices, a device in accordance with the present invention either uses less RAM and is thus less expensive, or runs faster using the same amount of RAM."

Surine, 2:49-55.

> "In accordance with the present invention, a set of high-use functions are decompressed out of ROM and instantiated in RAM using a patch manager."

Surine, 3:7-9.

> "In this embodiment, in addition to instantiating certain high-use functions, a memory configuration manager decompresses new software functions out of ROM, instantiates them in RAM, and updates the function pointer table to link them dynamically, as the capability of the new software functions are needed."

Surine, 3:27-32.

> "Additionally, the system of the present invention maintains the speed and responsiveness of the device while reducing the amount of RAM used in the device. In comparison to prior art embedded system devices, a device in accordance with the present invention either uses less RAM and is thus less expensive, or runs faster using the same amount of RAM."

Surine, 3:39-45.

> "In accordance with the present invention, the size of RAM 315 is minimized by running the majority of the software of embedded system 300 (e.g., boot code 401 and operating system code 403) from ROM 310. However, the overall speed of system 300 is largely maintained by running the most frequently-used software (e.g., high-use functions 416) from RAM 315"

Surine, 5:24-30.

> "Patch manager 405 subsequently executes. Patch manager 405 includes decompression software which decompresses compressed high-use functions 402 and, as described in greater detail in the discussion of FIG. 5 below, loads the resulting decompressed high-use functions 416 into RAM 315."

Surine, 5:35-40.

> "Consequently, since RAM 315 is accessed at least three times faster that ROM 310, system 300 runs much faster than a prior art embedded system running the entirety of its operating system code from ROM. The majority of RAM 315 is occupied by capture buffer 414, display buffer 415, and working memory 420. By using a relatively small portion of RAM 315 to run the uncompressed high-use functions 416, the speed of computer system 300 is greatly increased."

Surine, 5:62-6:3.

> "Patch manager 405 then loads the decompressed high-use functions 416 into RAM 315 and updates function pointer table 510 with entries for high-use functions 416."

Surine, 6:32-34. *See also* Surine, 7:5-43, 8:14-17, 8:36-43.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Surine, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Surine discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "Similarly, in addition to extending or modifying functions of operating system code 403, patch manager 405 can extend the functionality of any software application which executes on embedded system 300."

Surine, 6:59-62.

> "After boot and system instantiation is complete, application code uses an available block of memory within RAM 315 to use as a display buffer 751, working memory 752, and a capture buffer 753."

Surine, 7:55-58.

> "In step 807, the initial software environment for embedded system 300 is set up by an application. For example, a first executed application (e.g., the default application code which executes after power-up) uses a block of available memory of RAM 315 for capture buffers, display buffers, working memory, and other software data structures which need to be instantiated in the writeable address space of RAM 314."

Surine, 8:63-9:3.

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Surine, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1.3, and 1.4 above.	

4. The method of claim 1, further comprising updating the list of boot data.	Surine, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Surine discloses this limitation:

See Claim 1.1 above.

See also

> "At least one high-use function is decompressed out of the non-volatile memory and instantiated in volatile memory. The function pointer table is updated using a patch manager to incorporate an entry for the high-use function(s). The operating system code is executed from the non-volatile memory while the high-use function is executed from the volatile memory."

Surine, Abstract.

> "The patch manager subsequently updates the function pointer table to incorporate an entry for the high-use functions, thereby linking the high-use functions with the rest of the instantiated functions."

Surine, 3:15-18.

> "Patch manager 405 then loads the decompressed high-use functions 416 into RAM 315 and updates function pointer table 510 with entries for high-use functions 416."

Surine: 6:32-34.

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Surine, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Surine, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Surine, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Surine, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1.2, 1.3, and 1.4 above.	

7.2 a cache memory device; and;	Surine, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Surine, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Surine, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Surine, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Surine, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Surine discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> "Similarly, in addition to extending or modifying functions of operating system code 403, patch manager 405 can extend the functionality of any software application which executes on embedded system 300."

Surine, 6:59-62.

> "After boot and system instantiation is complete, application code uses an available block of memory within RAM 315 to use as a display buffer 751, working memory 752, and a capture buffer 753."

Surine, 7:55-58.

> "In step 807, the initial software environment for embedded system 300 is set up by an application. For example, a first executed application

(e.g., the default application code which executes after power-up) uses a block of available memory of RAM 315 for capture buffers, display buffers, working memory, and other software data structures which need to be instantiated in the writeable address space of RAM 314."

Surine, 8:63-9:3.

9.1 maintaining a list of application data associated with an application program;	Surine, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Surine, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Surine, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Surine discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> "Similarly, in addition to extending or modifying functions of operating system code 403, patch manager 405 can extend the functionality of any software application which executes on embedded system 300."

Surine, 6:59-62.

> "After boot and system instantiation is complete, application code uses an available block of memory within RAM 315 to use as a display buffer 751, working memory 752, and a capture buffer 753."

Surine, 7:55-58.

> "In step 807, the initial software environment for embedded system 300 is set up by an application. For example, a first executed application (e.g., the default application code which executes after power-up) uses a block of available memory of RAM 315 for capture buffers, display

buffers, working memory, and other software data structures which need to be instantiated in the writeable address space of RAM 314."

Surine, 8:63-9:3.

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Surine, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Surine discloses this limitation:

> "At power-up, boot code stored in the non-volatile memory is executed and begins instantiating the initial operating environment of the embedded computer system. A function pointer table is instantiated in the volatile memory, wherein the function pointer table includes a plurality of entries for a corresponding plurality of instantiated functions, wherein at least one entry is for operating system code stored in the non-volatile memory."

Surine, Abstract.

> "At power-up, boot code stored in the ROM is executed and begins instantiating the initial operating environment of the embedded system. A function pointer table is instantiated in the RAM. The function pointer table has entries, or function pointers, for each instantiated function such that they can each call each other and pass execution. The function pointer table has entries for functions which are instantiated in ROM and entries for functions which are instantiated in RAM."

Surine, 2:66-3:7.

> "Referring now to FIG. 4, a memory diagram depicting the contents of ROM 310 and RAM 315 is shown. As shown in FIG. 4, ROM 310 stores software including boot code 401, compressed high-use functions 402, and operating system code 403."

Surine, 5:15-19.

> "Function pointer table 510 includes a plurality of entries, or function pointers, which, when called by processing unit 305, redirect program execution to the memory address of a selected routine. The function pointers of function pointer table allow instantiated functions, whether executing from ROM 310 or RAM 315, to call one another. Initially, the function pointers are initialized to addresses in ROM 310, but after copying to RAM 315, the high-use function pointers are updated to addresses in RAM 315."
>
> Surine, 6:24-29. *See also* Surine, 7:5-21, 8:54-56.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Surine, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Surine discloses this limitation:

> "At least one high-use function is decompressed out of the non-volatile memory and instantiated in volatile memory. The function pointer table is updated using a patch manager to incorporate an entry for the high-use function(s). The operating system code is executed from the non-volatile memory while the high-use function is executed from the volatile memory."

Surine, Abstract.

> "In accordance with the present invention, a set of high-use functions are decompressed out of ROM and instantiated in RAM using a patch manager."

Surine, 3:7-9. *See also* 8:14-17, 8:36-43.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Surine, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Surine, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Surine, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Surine, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Surine discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

> "Patch manager 405 includes decompression software which decompresses compressed high-use functions 402 and, as described in greater detail in the discussion of FIG. 5 below, loads the resulting decompressed high-use functions 416 into RAM 315. This is shown by arrow 410."

Surine, 5:36-41.

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Surine, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Surine, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above.	

22.1 maintaining a list of boot data used for booting a computer system;.	Surine, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claim 1.1 above.	

22.2 initializing a central processing unit of the computer system;.	Surine, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claim 1.2 above.	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Surine, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claim 1.3 above.	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Surine, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claim 1.4 above.	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Surine, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 4, 10 and 11 above.	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Surine, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 15 and 22 above.	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Surine, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 16 and 22 above.	

27.1 a boot device..	Surine, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claim 1 above.	

27.2 a processor..	Surine, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claim 1.2 above.	

27.3 cache memory; and.	Surine, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 1.3 and 1.4 above.	

27.4 non-volatile memory for storing logic code for use by the processor,..	Surine, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claim 1.1 and 1.3 above.	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Surine, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claim 1.1 above.	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Surine, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claim 1.3 above.	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Surine, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claim 1.4 above.	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Surine, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 4, 10, and 11 above.	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Surine, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 15 and 27 above.	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Surine, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Surine discloses this limitation: *See* Claims 16 and 27 above.	

Appendix A22
Invalidity of U.S. Patent 7,181,608 based on Teoman

U.S. Patent No. 6,370,614 Teoman ("Teoman") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A22
Invalidity of U.S. Patent 7,181,608 based on Teoman

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Teoman, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Teoman discloses this limitation:

> "In one embodiment, the user-cache is non-volatile and large enough to hold several hundred megabytes worth of data. Consequently, by configuring the user-cache to store program code and configuration information used for computer system startup, the user-cache may be used as a boot source to provide much faster system boot up than can be achieved by booting out of mass storage media such as disk or tape."

Teoman, 3:10-17. *See also* Teoman, 3:18-45.

> "According to one embodiment, the user cache 25 includes boot firmware 66 for storing program code that is used to support operation of the user cache at system startup."

Teoman, 7:50-52.

> "The user interface 123 also permits the user to configure the computer system to use the user cache as a boot device, meaning that the user cache will be treated as a source of boot software at system startup. In one embodiment, if the user enables the "Use as a boot device" option, the user cache manager process prompts the user to specify the logical drive that is ordinarily the source of boot software. The user cache manager software responds by interacting with the BIOS configuration to treat the user-cache as the user-specified logical drive and thereby to boot out of the user-cache at system startup."

Teoman, 13:29-39.

> "In another embodiment, boot program code in the user cache firmware is executed to redirect boot time I/O requests before the operating system is loaded. In this way, boot time I/O requests are redirected to the user

> cache, making the user cache operable as a source of boot files during the entire boot sequence."

Teoman, 13:46-51.

> "In yet another embodiment for using the user cache to support system startup, before the OS boot sequence is begun, boot program code in the user cache firmware is executed to notify the system BIOS that the user cache is a bootable mass storage device. Also, when executed, the boot program code loads software into system memory for operating the user cache as a bootable mass storage device. In this embodiment, the user selects the user cache as the boot device in the system BIOS settings. Then, during the boot sequence, the operating system accesses boot up files in the user cache."

Teoman, 13:52-62.

Appendix A22
Invalidity of U.S. Patent 7,181,608 based on Teoman

1.1 maintaining a list of boot data used for booting a computer system;	Teoman, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Teoman discloses this limitation:

> "According to one embodiment, the user cache 25 includes boot firmware 66 for storing program code that is used to support operation of the user cache at system startup."

Teoman, 7:50-52.

> "In another embodiment, boot program code in the user cache firmware is executed to redirect boot time I/O requests before the operating system is loaded. In this way, boot time I/O requests are redirected to the user cache, making the user cache operable as a source of boot files during the entire boot sequence."

Teoman, 13:46-51.

> "In yet another embodiment for using the user cache to support system startup, before the OS boot sequence is begun, boot program code in the user cache firmware is executed to notify the system BIOS that the user cache is a bootable mass storage device. Also, when executed, the boot program code loads software into system memory for operating the user cache as a bootable mass storage device. In this embodiment, the user selects the user cache as the boot device in the system BIOS settings. Then, during the boot sequence, the operating system accesses boot up files in the user cache."

Teoman, 13:52-62.

1.2 initializing a central processing unit of the computer system;	Teoman, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Teoman discloses this limitation:

> "The program code is returned to the bus interface circuitry 61 which outputs it to the expansion bus where it is routed to its ultimate destination (e.g., the processor used to execute system boot code)."

Teoman, 7:61-64.

> "Still referring to FIG. 1, the processing unit 12 includes one or more processors that fetch program code from system memory 16 and execute the code to operate on data and to read and Write data to the system memory 16 and to the I/O devices on the expansion bus 18."

Teoman, 4:8-12.

> "In another embodiment, boot program code in the user cache firmware is executed to redirect boot time I/O requests before the operating system is loaded. In this way, boot time I/O requests are redirected to the user cache, making the user cache operable as a source of boot files during the entire boot sequence."

Teoman, 13:46-51.

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Teoman, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Teoman discloses this limitation:

> "When an I/O request 40 to access the mass storage 46 is issued (e.g., a file read or write request issued in the course of executing an application program), the I/O request 40 is first applied to the OS cache maintained in system memory 16. If the I/O request 40 hits the OS cache (i.e., the data sought to be accessed is cached in the OS cache), the access is performed in the OS cache. If the I/O request 40 is a read request, the data is returned to the requestor. If the I/O request 40 does not hit the OS cache, the I/O request 40 is redirected from the mass storage 46 to the user cache 25 by software mechanisms described below. If the I/O request 40 hits the user cache 25, the access is performed in the user cache 25 without having to access the mass storage 46, thereby substantially reducing the overall access time. Also, because the user cache 25 is significantly larger than the OS cache and supports data preloading (discussed below), much higher hit rates can be achieved in the user cache than in the OS cache."

Teoman, 4:57-5:7. *See also* Teoman, 5:16-20, 5:48-51.

> "According to one embodiment, the user cache 25 includes boot firmware 66 for storing program code that is used to support operation of the user cache at system startup."

Teoman, 7:50-52.

> "In general, there are two types of storage operations that take place in

> the user cache: preloading and responsive caching. Responsive caching refers to the storage of data in the user cache in response to I/O requests from application processes. In a preload operation, by contrast, data is retrieved from mass storage and stored in the user cache before being requested for use in an application process."

Teoman, 9:24-31. *See also* Teoman, 9:31-44, 10:40-60.

> "The user interface 123 also permits the user to configure the computer system to use the user cache as a boot device, meaning that the user cache will be treated as a source of boot software at system startup. In one embodiment, if the user enables the "Use as a boot device" option, the user cache manager process prompts the user to specify the logical drive that is ordinarily the source of boot software. The user cache manager software responds by interacting with the BIOS configuration to treat the user-cache as the user-specified logical drive and thereby to boot out of the user-cache at system startup."

Teoman, 13:29-39.

> "In another embodiment, the user cache driver is loaded very early in the OS boot sequence and is operable for most of the sequence. Most operating systems support loading device drivers early in the boot sequence so that, in most cases, this mode of operation requires no special hardware or software support."

Teoman, 13:40-45.

> "In yet another embodiment for using the user cache to support system startup, before the OS boot sequence is begun, boot program code in the user cache firmware is executed to notify the system BIOS that the user cache is a bootable mass storage device. Also, when executed, the boot program code loads software into system memory for operating the user cache as a bootable mass storage device. In this embodiment, the user selects the user cache as the boot device in the system BIOS settings. Then, during the boot sequence, the operating system accesses boot up files in the user cache."

Teoman, 13:52-62.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Teoman, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Teoman discloses this limitation:

> "When an I/O request 40 to access the mass storage 46 is issued (e.g., a file read or write request issued in the course of executing an application program), the I/O request 40 is first applied to the OS cache maintained in system memory 16. If the I/O request 40 hits the OS cache (i.e., the data sought to be accessed is cached in the OS cache), the access is performed in the OS cache. If the I/O request 40 is a read request, the data is returned to the requestor. If the I/O request 40 does not hit the OS cache, the I/O request 40 is redirected from the mass storage 46 to the user cache 25 by software mechanisms described below. If the I/O request 40 hits the user cache 25, the access is performed in the user cache 25 without having to access the mass storage 46, thereby substantially reducing the overall access time. Also, because the user cache 25 is significantly larger than the OS cache and supports data preloading (discussed below), much higher hit rates can be achieved in the user cache than in the OS cache."

Teoman, 4:57-5:7. *See also* Teoman, 5:16-20, 5:48-51.

> "After a user has specified a set of commanded preload parameters, the

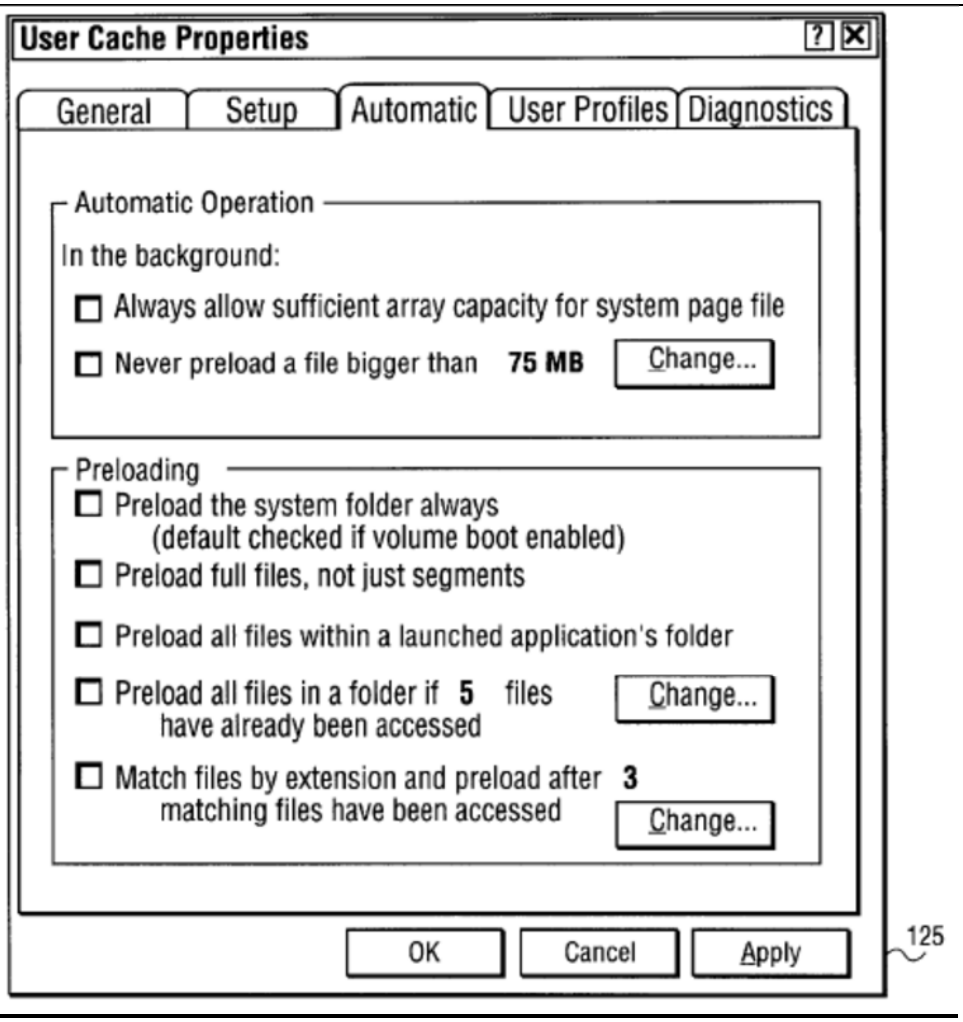
> user cache manager 90 responds by generating I/O requests to retrieve the data identified by the preload parameters from mass storage 46"

Teoman, 9:56-59.

> "In yet another embodiment for using the user cache to support system startup, before the OS boot sequence is begun, boot program code in the user cache firmware is executed to notify the system BIOS that the user cache is a bootable mass storage device. Also, when executed, the boot program code loads software into system memory for operating the user cache as a bootable mass storage device. In this embodiment, the user selects the user cache as the boot device in the system BIOS settings. Then, during the boot sequence, the operating system accesses boot up files in the user cache."

Teoman, 13:52-62.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Teoman, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above. *See also*	



Teoman, Fig. 10.

> "In another embodiment, an application program called a user cache manager is executed to receive user preferences as to what data to store and not to store in the user cache."

Teoman, 3:18-20.

> "The program code in the system memory 16 includes operating system (OS) program code 30, application program code 34 and device driver program code 32. The application program code 34 is executed by the processing unit 12 to implement application processes which, in turn, invoke operating system services to display user-interfaces, operate on user-input and access user-specified data."

Teoman, 4:16-22.

Teoman Claim 2

"The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

> "For example, when an application process invokes an operating system service to perform I/O to a device attached to the expansion bus 18, the operating system 30 invokes a standard routine Within the appropriate device driver 32 to carry out the requested I/O."

Teoman, 4:35-40.

> "When an I/O request 40 to access the mass storage 46 is issued (e.g., a ?le read or write request issued in the course of executing an application program), the I/O request 40 is first applied to the OS cache maintained in system memory 16."

Teoman, 4:57-61.

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Teoman, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Teoman discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

> "The expansion bus 18 supports connection of a number of I/O devices including a self-buffered disk drive controller 20 and its associated disk drive 26, a non-buffered disk drive controller 22 and its associated disk drive 28, a network access device 24 such as a modem or local/Wide area network communications card and a user cache 25."

Teoman, 3:51-57.

> "The user-cache 25 includes bus inter face circuitry 61, a DRAM controller 63 and a plurality of rows (1 through N) of DRAM components 68A—68C, 69A—69C, 70A—70C."

Teoman, 6:16-19.

> "The address and control inputs of the DRAM components within a given row are coupled to a common address path and a common control path from the DRAM controller 63."

Teoman, 6:24-27.

> "In one embodiment, the power source selector 67 distinguishes between full system power and trickle power and asserts a sleep signal 81 to the DRAM controller 63 Whenever the user cache is being powered by trickle power or battery power. The DRAM controller 63 responds to the sleep signal 81 by issuing control signals to place the DRAM components of the user cache 25 in reduced power state. In the

reduced power state, DRAM refresh operations are continued either under control of the DRAM controller 63 or by logic on board the DRAM components themselves. Other logic elements Within the user-cache 25, including the bus interface circuitry 61 and portions of the DRAM controller that operate on access requests from the bus interface circuitry are shut down to save power."

Teoman, 7:18-32.

4. The method of claim 1, further comprising updating the list of boot data.	Teoman, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Teoman discloses this limitation:

See Claim 1.1 above.

> "The requested blocks are then passed back to the user cache driver 45 which writes them to the user cache 25 and updates the user cache directory."

Teoman, 10:20-23.

> "After loading data into the user cache 25, the user cache driver 45 updates the user cache directory to indicate the newly cached blocks."

Teoman, 12:50-52.

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Teoman, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Teoman, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Teoman, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Teoman, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Teoman discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

> "The expansion bus 18 supports connection of a number of I/O devices including a self-buffered disk drive controller 20 and its associated disk drive 26, a non-buffered disk drive controller 22 and its associated disk drive 28, a network access device 24 such as a modem or local/Wide area network communications card and a user cache 25."

Teoman, 3:51-57.

> "The user-cache 25 includes bus inter face circuitry 61, a DRAM controller 63 and a plurality of rows (1 through N) of DRAM components 68A—68C, 69A—69C, 70A—70C."

Teoman, 6:16-19.

> "The address and control inputs of the DRAM components within a given row are coupled to a common address path and a common control path from the DRAM controller 63."

Teoman, 6:24-27.

> "In one embodiment, the power source selector 67 distinguishes between full system power and trickle power and asserts a sleep signal 81 to the DRAM controller 63 Whenever the user cache is being powered by trickle power or battery power. The DRAM controller 63 responds to the sleep signal 81 by issuing control signals to place the DRAM components of the user cache 25 in reduced power state. In the reduced power state, DRAM refresh operations are continued either under control of the DRAM controller 63 or by logic on board the

DRAM components themselves. Other logic elements Within the user-cache 25, including the bus interface circuitry 61 and portions of the DRAM controller that operate on access requests from the bus interface circuitry are shut down to save power."

Teoman, 7:18-32.

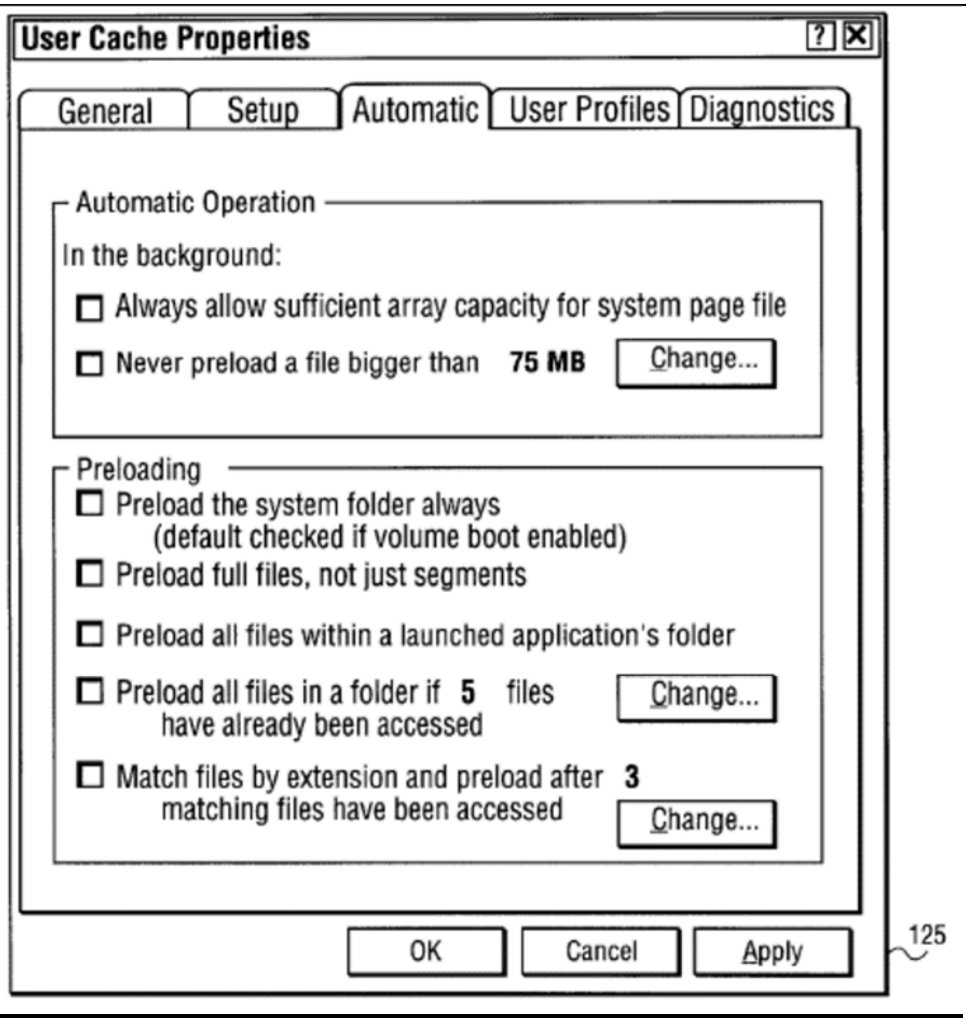
7.2 a cache memory device; and;	Teoman, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Teoman, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Teoman, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Teoman, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Teoman, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.1, 1.3, 1.4, 2, 3, and 7 above. *See also*	



Teoman, Fig. 10.

> "In another embodiment, an application program called a user cache manager is executed to receive user preferences as to what data to store and not to store in the user cache."

Teoman, 3:18-20.

> "The program code in the system memory 16 includes operating system (OS) program code 30, application program code 34 and device driver program code 32. The application program code 34 is executed by the processing unit 12 to implement application processes which, in turn, invoke operating system services to display user-interfaces, operate on user-input and access user-specified data."

Teoman, 4:16-22.

> "For example, when an application process invokes an operating system service to perform I/O to a device attached to the expansion bus 18, the operating system 30 invokes a standard routine Within the appropriate device driver 32 to carry out the requested I/O."

Teoman, 4:35-40.

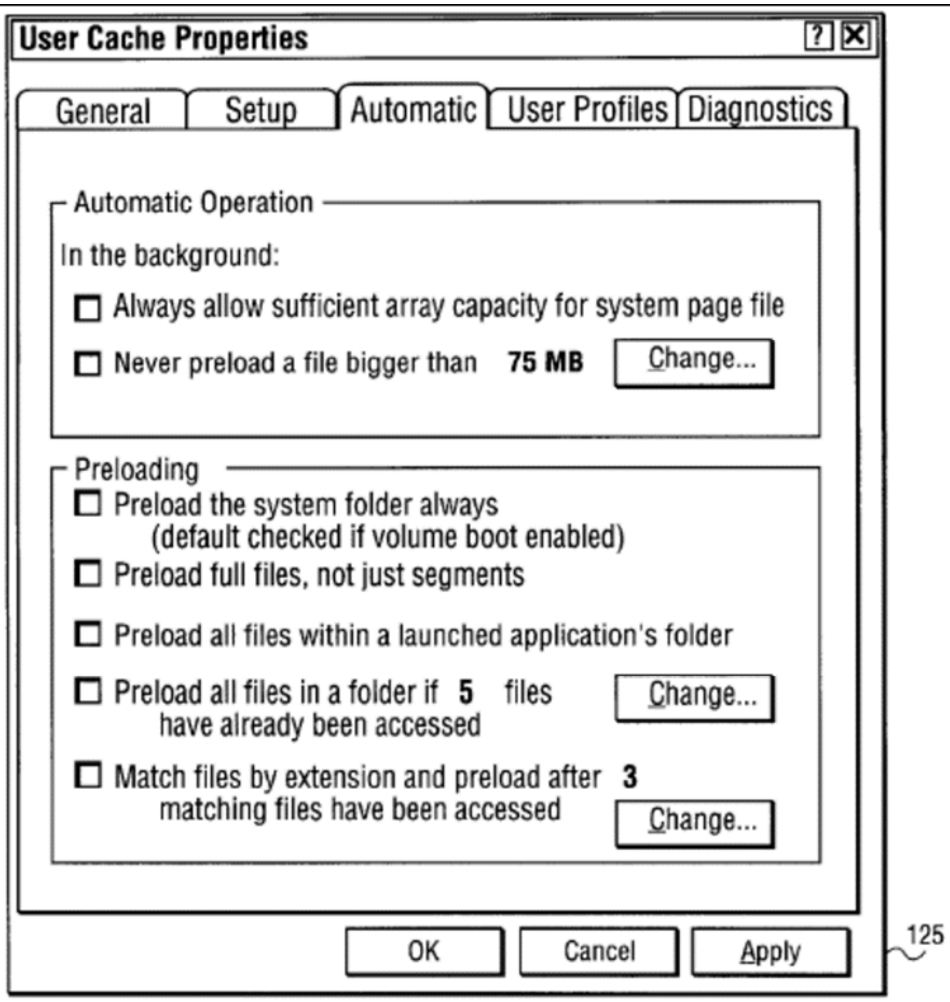
> "When an I/O request 40 to access the mass storage 46 is issued (e.g., a ?le read or write request issued in the course of executing an application program), the I/O request 40 is first applied to the OS cache maintained in system memory 16."

Teoman, 4:57-61.

9.1 maintaining a list of application data associated with an application program;	Teoman, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Teoman, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Teoman, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.4, 2, and 8 above. *See also* ***Look for additional references to application programs***	



Teoman, Fig. 10.

> "In another embodiment, an application program called a user cache manager is executed to receive user preferences as to what data to store and not to store in the user cache."

Teoman, 3:18-20.

> "The program code in the system memory 16 includes operating system (OS) program code 30, application program code 34 and device driver program code 32. The application program code 34 is executed by the processing unit 12 to implement application processes which, in turn, invoke operating system services to display user-interfaces, operate on user-input and access user-specified data."

Teoman, 4:16-22.

> "For example, when an application process invokes an operating system service to perform I/O to a device attached to the expansion bus 18, the operating system 30 invokes a standard routine Within the appropriate device driver 32 to carry out the requested I/O."

Teoman, 4:35-40.

> "When an I/O request 40 to access the mass storage 46 is issued (e.g., a ?le read or write request issued in the course of executing an application program), the I/O request 40 is first applied to the OS cache maintained in system memory 16."

Teoman, 4:57-61.

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Teoman, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Teoman discloses this limitation:

See ***Copy from 936 element 1.1***

> "According to one embodiment, the user cache 25 includes boot firmware 66 for storing program code that is used to support operation of the user cache at system startup."

Teoman, 7:50-52.

> "In yet another embodiment for using the user cache to support system startup, before the OS boot sequence is begun, boot program code in the user cache firmware is executed to notify the system BIOS that the user cache is a bootable mass storage device. Also, when executed, the boot program code loads software into system memory for operating the user cache as a bootable mass storage device. In this embodiment, the user selects the user cache as the boot device in the system BIOS settings. Then, during the boot sequence, the operating system accesses boot up files in the user cache."

Teoman, 13:52-62.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Teoman, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Teoman, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Teoman, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Teoman, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Teoman, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.3, 1.4, and 15 above.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Teoman, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Teoman, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above.	

22.1 maintaining a list of boot data used for booting a computer system;.	Teoman, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claim 1.1 above.	

22.2 initializing a central processing unit of the computer system;.	Teoman, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claim 1.2 above.	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Teoman, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claim 1.3 above.	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Teoman, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claim 1.4 above.	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Teoman, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 4, 10 and 11 above.	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Teoman, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 15 and 22 above.	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Teoman, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 16 and 22 above.	

27.1 a boot device..	Teoman, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claim 1 above.	

27.2 a processor..	Teoman, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claim 1.2 above.	

27.3 cache memory; and.	Teoman, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 1.3 and 1.4 above.	

27.4 non-volatile memory for storing logic code for use by the processor,..	Teoman, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claim 1.1 and 1.3 above.	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Teoman, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claim 1.1 above.	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Teoman, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claim 1.3 above.	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Teoman, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claim 1.4 above.	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Teoman, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 4, 10, and 11 above.	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Teoman, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 15 and 27 above.	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Teoman, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Teoman discloses this limitation: *See* Claims 16 and 27 above.	

Appendix A23
Invalidity of U.S. Patent 7,181,608 based on Vers

German Patent No. DE19721786 to Michael Vers ("Vers") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A23
Invalidity of U.S. Patent 7,181,608 based on Vers

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Vers, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Vers discloses this limitation:

> The method involves placing system software into a non-volatile memory (16) in compressed form. After a reset, the software is copied into the volatile memory (18) and is started there. This is done either after decompression or without decompression, depending on the characteristic element. Data corresponding to changes in the user software prior to voltage drop-out are compressed and stored in the non-volatile memory and after reset are decompressed for regeneration of the data in volatile memory.

Vers, Abstract

> From the prior art it is known that, in a data processing apparatus, both the system and the application software is stored in a nonvolatile memory. The system software can run on the one hand directly from the non-volatile memory or on the other hand copied from the nonvolatile memory to the volatile memory and run from this. The start of the system software from non-volatile memory has the disadvantage that either a quicker and thereby more expensive non-volatile memory must be used, or that the system performance is degraded by inserting wait states. At the start of the system software from the volatile memory, a nonvolatile memory full size of the system software must be available, but which is no longer needed after the copying of the system software.

Vers, 1

> The invention is based on the problem to provide a method for operating a data processing device in such a way that non-volatile at least consistent performance storage elements with a minimum memory capacity can be used. the problem is inventively solved in that the system software stored in compressed form in the non-volatile memory and to reset in response

to a flag element either is decompressed and stored in the volatile memory or copied without decompression in the volatile memory and started there and / or that the application software associated, are compressed changes of application software files containing or by instructing from the programming device before power failure and stored in the nonvolatile memory and decompressed when power is restored to regenerate the files in the volatile memories.

Vers, 2

Preferably, a destination address in the volatile memory is in the header in the non-volatile memory, where the system software such as operating system copied or stored decompressed. It is also provided that after decompression or copying of the system software calculated CRC checksum compared with a in a second header in the region of the beginning of the file system software in the volatile memory, and if they match a start routine is started.

Vers, 2

1.1 maintaining a list of boot data used for booting a computer system;	Vers, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Vers discloses this limitation:

> The method involves placing system software into a non-volatile memory (16) in compressed form. After a reset, the software is copied into the volatile memory (18) and is started there. This is done either after decompression or without decompression, depending on the characteristic element. Data corresponding to changes in the user software prior to voltage drop-out are compressed and stored in the non-volatile memory and after reset are decompressed for regeneration of the data in volatile memory.

Vers, Abstract

> Preferably, a destination address in the volatile memory is in the header in the non-volatile memory, where the system software such as operating system copied or stored decompressed. It is also provided that after decompression or copying of the system software calculated CRC checksum compared with a in a second header in the region of the beginning of the file system software in the volatile memory, and if they match a start routine is started.

Vers, 2

> In Fig. 1 the basic construction of a programmable controller is shown with a bus 12 to which a microprocessor 14, non-volatile and volatile memories 16, 18 and one or more peripheral devices 20 are connected. as non-volatile memory 16 preferably flash memory modules are used, which can be written and read, and in contrast to the volatile memory 18 may be retained their content when they are not supplied with operating voltage. Other non-volatile memory such as EPROM and EEPROM can also be used. Preferably as a volatile memory SRAM and / or DRAM's are used. In Fig. 2, the memory map of the nonvolatile memory 16 is shown. According to the invention, that the system software as the

operating system stored in an initial region 22nd Following the operating system is followed by a free space 24, which is a storage area 26 connects with a reset initialization and a decompression algorithm as executable code. Especially with programmable logic controllers or PLCs can also use the software in the nonvolatile memory is stored in compressed form be.

Vers, 3

By a further storage area 36, the destination address is specified in the volatile memory to which the operating system copies or store decompressed. Finally, the tray of the operating system begins in compressed or transparent form.

Vers, 3

1.2 initializing a central processing unit of the computer system;	Vers, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Vers discloses this limitation:

> The method involves placing system software into a non-volatile memory (16) in compressed form. After a reset, the software is copied into the volatile memory (18) and is started there. This is done either after decompression or without decompression, depending on the characteristic element. Data corresponding to changes in the user software prior to voltage drop-out are compressed and stored in the non-volatile memory and after reset are decompressed for regeneration of the data in volatile memory.

Vers, Abstract

> Preferably, a destination address in the volatile memory is in the header in the non-volatile memory, where the system software such as operating system copied or stored decompressed. It is also provided that after decompression or copying of the system software calculated CRC checksum compared with a in a second header in the region of the beginning of the file system software in the volatile memory, and if they match a start routine is started.

Vers, 2

> In Fig. 1 the basic construction of a programmable controller is shown with a bus 12 to which a microprocessor 14, non-volatile and volatile memories 16, 18 and one or more peripheral devices 20 are connected. as non-volatile memory 16 preferably flash memory modules are used, which can be written and read, and in contrast to the volatile memory 18 may be retained their content when they are not supplied with operating voltage. Other non-volatile memory such as EPROM and EEPROM can also be used. Preferably as a volatile memory SRAM and / or DRAM's are used. In Fig. 2, the memory map of the nonvolatile memory 16 is shown. According to the invention, that the system software as the

operating system stored in an initial region 22nd Following the operating system is followed by a free space 24, which is a storage area 26 connects with a reset initialization and a decompression algorithm as executable code. Especially with programmable logic controllers or PLCs can also use the software in the nonvolatile memory is stored in compressed form be. Vers, 3

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Vers, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Vers discloses this limitation:

> The method involves placing system software into a non-volatile memory (16) in compressed form. After a reset, the software is copied into the volatile memory (18) and is started there. This is done either after decompression or without decompression, depending on the characteristic element. Data corresponding to changes in the user software prior to voltage drop-out are compressed and stored in the non-volatile memory and after reset are decompressed for regeneration of the data in volatile memory.

Vers, Abstract

> The invention is based on the problem to provide a method for operating a data processing device in such a way that non-volatile at least consistent performance storage elements with a minimum memory capacity can be used. the problem is inventively solved in that the system software stored in compressed form in the non-volatile memory and to reset in response to a flag element either is decompressed and stored in the volatile memory or copied without decompression in the volatile memory and started there and / or that the application software associated, are compressed changes of application software files containing or by instructing from the programming device before power failure and stored in the nonvolatile memory and decompressed when power is restored to regenerate the files

in the volatile memories.

Vers, 2

Preferably, a destination address in the volatile memory is in the header in the non-volatile memory, where the system software such as operating system copied or stored decompressed. It is also provided that after decompression or copying of the system software calculated CRC checksum compared with a in a second header in the region of the beginning of the file system software in the volatile memory, and if they match a start routine is started.

Vers, 2

A particularly preferred compressing algorithm is the LZW algorithm used.

Vers, 2

In Fig. 1 the basic construction of a programmable controller is shown with a bus 12 to which a microprocessor 14, non-volatile and volatile memories 16, 18 and one or more peripheral devices 20 are connected. as non-volatile memory 16 preferably flash memory modules are used, which can be written and read, and in contrast to the volatile memory 18 may be retained their content when they are not supplied with operating voltage. Other non-volatile memory such as EPROM and EEPROM can also be used. Preferably as a volatile memory SRAM and / or DRAM's are used. In Fig. 2, the memory map of the nonvolatile memory 16 is shown. According to the invention, that the system software as the operating system stored in an initial region 22nd Following the operating system is followed by a free space 24, which is a storage area 26 connects with a reset initialization and a decompression algorithm as executable code. Especially with programmable logic controllers or PLCs can also use the software in the nonvolatile memory is stored in compressed form be.

Vers, 3

By a further storage area 36, the destination address is specified in the volatile memory to which the operating system copies or store decompressed. Finally, the tray of the operating system begins in compressed or transparent form.

Vers, 3

To inventively to guarantee the use of non-volatile memories of a minimum memory size is provided that the stored in compressed form in the non-volatile memory operating system during initialization in the volatile memory is loaded. For this purpose it is provided that the non-volatile memory compresses stored operating system, ie, when the flag element the value "ONE" has a function of the marker element, decompressed by a stored in nonvolatile memory uncompressed routine and is stored in the nonvolatile memory. Before the decompression of the operating system a CRC checksum is calculated and compared with the likewise in the header 28 stored in the storage area 30 by means of a checksum stored in the memory area 32 of the first header 28 length information. Is the calculated checksum matches the checksum stored, the process continues with the decompression. Here, the operating system is decompressed without the header 28 and stored decompressed to the stored in the memory area 36 of the header 28 destination address in the volatile memory. Does the marker element the value "zero" on, is the operating system directly, ie without decompression and without header in the volatile memory copied. After decompress or copying of the operating system as previously described a CRC checksum is determined and compared with a in a second header 40 in volatile memory which is present in decompressed form. If the two checksums match, a startup routine of the operating system is started, whose address is also stored in the header 40 in the storage area 46th

Vers, 3-4

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Vers, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Vers discloses this limitation:

> The method involves placing system software into a non-volatile memory (16) in compressed form. After a reset, the software is copied into the volatile memory (18) and is started there. This is done either after decompression or without decompression, depending on the characteristic element. Data corresponding to changes in the user software prior to voltage drop-out are compressed and stored in the non-volatile memory and after reset are decompressed for regeneration of the data in volatile memory.

Vers, Abstract

> The invention is based on the problem to provide a method for operating a data processing device in such a way that non-volatile at least consistent performance storage elements with a minimum memory capacity can be used. the problem is inventively solved in that the system software stored in compressed form in the non-volatile memory and to reset in response to a flag element either is decompressed and stored in the volatile memory or copied without decompression in the volatile memory and started there and / or that the application software associated, are compressed changes

of application software files containing or by instructing from the programming device before power failure and stored in the nonvolatile memory and decompressed when power is restored to regenerate the files in the volatile memories.

Vers, 2

Preferably, a destination address in the volatile memory is in the header in the non-volatile memory, where the system software such as operating system copied or stored decompressed. It is also provided that after decompression or copying of the system software calculated CRC checksum compared with a in a second header in the region of the beginning of the file system software in the volatile memory, and if they match a start routine is started.

Vers, 2

A particularly preferred compressing algorithm is the LZW algorithm used.

Vers, 2

In Fig. 1 the basic construction of a programmable controller is shown with a bus 12 to which a microprocessor 14, non-volatile and volatile memories 16, 18 and one or more peripheral devices 20 are connected. as non-volatile memory 16 preferably flash memory modules are used, which can be written and read, and in contrast to the volatile memory 18 may be retained their content when they are not supplied with operating voltage. Other non-volatile memory such as EPROM and EEPROM can also be used. Preferably as a volatile memory SRAM and / or DRAM's are used. In Fig. 2, the memory map of the nonvolatile memory 16 is shown. According to the invention, that the system software as the operating system stored in an initial region 22nd Following the operating system is followed by a free space 24, which is a storage area 26 connects with a reset initialization and a decompression algorithm as executable code. Especially with programmable logic controllers or PLCs can also use the software in the nonvolatile memory is stored in compressed form be.

Vers, 3

By a further storage area 36, the destination address is specified in the volatile memory to which the operating system copies or store decompressed. Finally, the tray of the operating system begins in

compressed or transparent form.

Vers, 3

To inventively to guarantee the use of non-volatile memories of a minimum memory size is provided that the stored in compressed form in the non-volatile memory operating system during initialization in the volatile memory is loaded. For this purpose it is provided that the non-volatile memory compresses stored operating system, ie, when the flag element the value "ONE" has a function of the marker element, decompressed by a stored in nonvolatile memory uncompressed routine and is stored in the nonvolatile memory. Before the decompression of the operating system a CRC checksum is calculated and compared with the likewise in the header 28 stored in the storage area 30 by means of a checksum stored in the memory area 32 of the first header 28 length information. Is the calculated checksum matches the checksum stored, the process continues with the decompression. Here, the operating system is decompressed without the header 28 and stored decompressed to the stored in the memory area 36 of the header 28 destination address in the volatile memory. Does the marker element the value "zero" on, is the operating system directly, ie without decompression and without header in the volatile memory copied. After decompress or copying of the operating system as previously described a CRC checksum is determined and compared with a in a second header 40 in volatile memory which is present in decompressed form. If the two checksums match, a startup routine of the operating system is started, whose address is also stored in the header 40 in the storage area 46th

Vers, 3-4

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Vers, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Vers discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> The invention relates to a method for operating a data processing device, in particular automation device, with volatile and non-volatile memories, said system and / or application software is copied from non-volatile to a volatile memory.

Vers, 1

> From the prior art it is known that in a data processing device, both the system and the application software is stored in a nonvolatile memory. The system software can run directly on the one hand from the non-volatile memory or on the other hand copies of the non-volatile memory to the volatile memory and run from this. The start of the system software from non-volatile memory has the disadvantage that either a quicker and thereby more expensive non-volatile memory must be used, or that the system power is discontinued by inserting wait states ago. At the start of the system software from the volatile memory non-volatile memory must be in full size of the system software, however, is no longer needed after copying the system software.

Vers, 1

> The problem is inventively achieved in that the system software stored compression form in the in the non-volatile memory after reset depending

on a flag element either is decompressed and stored in the volatile memory or copied without decompression in the volatile memory and started there and / or that the application software associated, are compressed files containing changes the user software or by appointment by the programmer before power failure and stored in non-volatile memory and decompressed when power is restored to regenerate the files in the volatile memories.

Vers 2

In particular, programmable logic controllers or PLCs can be stored in non-volatile memory in compressed form and application software.

Vers, 3

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Vers, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Vers discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

> In particular, programmable logic controllers or PLCs can be stored in non-volatile memory in compressed form and application software.

Vers, 3

> To not use volatile memory with a small storage capacity possible is, the invention proposes that an art compression routine at Beauf transfer compressed by a programmer the latch and the HEAP a memory management and stored in the nonvolatile memory 16 and when the PLC again is turned on, a Dekomprimierroutine is activated, which regenerates the signal memory, and the entire heap of memory management of the nonvolatile memory to the volatile memory. Stay All online changes obtained so that a further change of user program when Spannungsaus case can be avoided. In addition, while minimizing the hardware expense, the functionality of the programmable controller expanded.

Vers 4

4. The method of claim 1, further comprising updating the list of boot data.	Vers, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Vers discloses this limitation:

See Claim 1.1 above.

See also

> The method involves placing system software into a non-volatile memory (16) in compressed form. After a reset, the software is copied into the volatile memory (18) and is started there. This is done either after decompression or without decompression, depending on the characteristic element. Data corresponding to changes in the user software prior to voltage drop-out are compressed and stored in the non-volatile memory and after reset are decompressed for regeneration of the data in volatile memory.

Vers, Abstract

> Preferably, a destination address in the volatile memory is in the header in the non-volatile memory, where the system software such as operating system copied or stored decompressed. It is also provided that after decompression or copying of the system software calculated CRC checksum compared with a in a second header in the region of the beginning of the file system software in the volatile memory, and if they match a start routine is started.

Vers, 2

> In Fig. 1 the basic construction of a programmable controller is shown with a bus 12 to which a microprocessor 14, non-volatile and volatile memories 16, 18 and one or more peripheral devices 20 are connected. as non-volatile memory 16 preferably flash memory modules are used, which can be written and read, and in contrast to the volatile memory 18 may be retained their content when they are not supplied with operating voltage. Other non-volatile memory such as EPROM and EEPROM can also be used. Preferably as a volatile memory SRAM and / or DRAM's

are used. In Fig. 2, the memory map of the nonvolatile memory 16 is shown. According to the invention, that the system software as the operating system stored in an initial region 22nd Following the operating system is followed by a free space 24, which is a storage area 26 connects with a reset initialization and a decompression algorithm as executable code. Especially with programmable logic controllers or PLCs can also use the software in the nonvolatile memory is stored in compressed form be.

Vers, 3

By a further storage area 36, the destination address is specified in the volatile memory to which the operating system copies or store decompressed. Finally, the tray of the operating system begins in compressed or transparent form.

Vers, 3

If a user software from the volatile memory operates 18 may at any time provide the required memory locations for storing online modifications available memory management. For an online modified user program is again even with reclosing of the automation device in the previously amended version is available, it must be stored before switching off the automation device in the non-volatile memory, so that all previous changes and variable initialization are stored.

Vers, 4

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Vers, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Vers, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 1.1 and 4 above.	

Appendix A23
Invalidity of U.S. Patent 7,181,608 based on Vers

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Vers, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Vers, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Vers discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

> In particular, programmable logic controllers or PLCs can be stored in non-volatile memory in compressed form and application software.

Vers, 3

> To not use volatile memory with a small storage capacity possible is, the invention proposes that an art compression routine at Beauf transfer compressed by a programmer the latch and the HEAP a memory management and stored in the nonvolatile memory 16 and when the PLC again is turned on, a Dekomprimierroutine is activated, which regenerates the signal memory, and the entire heap of memory management of the nonvolatile memory to the volatile memory. Stay All online changes obtained so that a further change of user program when Spannungsaus case can be avoided. In addition, while minimizing the hardware expense, the functionality of the programmable controller expanded.

Vers 4

7.2 a cache memory device; and;	Vers, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Vers, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Vers, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Vers, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Vers, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Vers discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> The invention relates to a method for operating a data processing device, in particular automation device, with volatile and non-volatile memories, said system and / or application software is copied from non-volatile to a volatile memory.

Vers, 1

> From the prior art it is known that in a data processing device, both the system and the application software is stored in a nonvolatile memory. The system software can run directly on the one hand from the non-volatile memory or on the other hand copies of the non-volatile memory to the volatile memory and run from this. The start of the system software from non-volatile memory has the disadvantage that either a quicker and thereby more expensive non-volatile memory must be used, or that the

system power is discontinued by inserting wait states ago. At the start of the system software from the volatile memory non-volatile memory must be in full size of the system software, however, is no longer needed after copying the system software.

Vers, 1

The problem is inventively achieved in that the system software stored compression form in the in the non-volatile memory after reset depending on a flag element either is decompressed and stored in the volatile memory or copied without decompression in the volatile memory and started there and / or that the application software associated, are compressed files containing changes the user software or by appointment by the programmer before power failure and stored in non-volatile memory and decompressed when power is restored to regenerate the files in the volatile memories.

Vers 2

In particular, programmable logic controllers or PLCs can be stored in non-volatile memory in compressed form and application software.

Vers, 3

9.1 maintaining a list of application data associated with an application program;	Vers, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Vers, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Vers, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Vers discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> The invention relates to a method for operating a data processing device, in particular automation device, with volatile and non-volatile memories, said system and / or application software is copied from non-volatile to a volatile memory.

Vers, 1

> From the prior art it is known that in a data processing device, both the system and the application software is stored in a nonvolatile memory. The system software can run directly on the one hand from the non-volatile memory or on the other hand copies of the non-volatile memory to the volatile memory and run from this. The start of the system software from non-volatile memory has the disadvantage that either a quicker and thereby more expensive non-volatile memory must be used, or that the system power is discontinued by inserting wait states ago. At the start of the system software from the volatile memory non-volatile memory must

be in full size of the system software, however, is no longer needed after copying the system software.

Vers, 1

The problem is inventively achieved in that the system software stored compression form in the in the non-volatile memory after reset depending on a flag element either is decompressed and stored in the volatile memory or copied without decompression in the volatile memory and started there and / or that the application software associated, are compressed files containing changes the user software or by appointment by the programmer before power failure and stored in non-volatile memory and decompressed when power is restored to regenerate the files in the volatile memories.

Vers 2

In particular, programmable logic controllers or PLCs can be stored in non-volatile memory in compressed form and application software.

Vers, 3

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Vers, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Vers discloses this limitation:

> The method involves placing system software into a non-volatile memory (16) in compressed form. After a reset, the software is copied into the volatile memory (18) and is started there. This is done either after decompression or without decompression, depending on the characteristic element. Data corresponding to changes in the user software prior to voltage drop-out are compressed and stored in the non-volatile memory and after reset are decompressed for regeneration of the data in volatile memory.

Vers, Abstract

> The invention is based on the problem to provide a method for operating a data processing device in such a way that non-volatile at least consistent performance storage elements with a minimum memory capacity can be used. the problem is inventively solved in that the system software stored in compressed form in the non-volatile memory and to reset in response to a flag element either is decompressed and stored in the volatile memory or copied without decompression in the volatile memory and started there and / or that the application software associated, are compressed changes of application software files containing or by instructing from the programming device before power failure and stored in the nonvolatile memory and decompressed when power is restored to regenerate the files in the volatile memories.

Vers, 2

> Preferably, a destination address in the volatile memory is in the header

in the non-volatile memory, where the system software such as operating system copied or stored decompressed. It is also provided that after decompression or copying of the system software calculated CRC checksum compared with a in a second header in the region of the beginning of the file system software in the volatile memory, and if they match a start routine is started.

Vers, 2

A particularly preferred compressing algorithm is the LZW algorithm used.

Vers, 2

In Fig. 1 the basic construction of a programmable controller is shown with a bus 12 to which a microprocessor 14, non-volatile and volatile memories 16, 18 and one or more peripheral devices 20 are connected. as non-volatile memory 16 preferably flash memory modules are used, which can be written and read, and in contrast to the volatile memory 18 may be retained their content when they are not supplied with operating voltage. Other non-volatile memory such as EPROM and EEPROM can also be used. Preferably as a volatile memory SRAM and / or DRAM's are used. In Fig. 2, the memory map of the nonvolatile memory 16 is shown. According to the invention, that the system software as the operating system stored in an initial region 22nd Following the operating system is followed by a free space 24, which is a storage area 26 connects with a reset initialization and a decompression algorithm as executable code. Especially with programmable logic controllers or PLCs can also use the software in the nonvolatile memory is stored in compressed form be.

Vers, 3

By a further storage area 36, the destination address is specified in the volatile memory to which the operating system copies or store decompressed. Finally, the tray of the operating system begins in compressed or transparent form.

Vers, 3

To inventively to guarantee the use of non-volatile memories of a minimum memory size is provided that the stored in compressed form in the non-volatile memory operating system during initialization in the volatile memory is loaded. For this purpose it is provided that the non-

Vers Claim 10

"The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device"

volatile memory compresses stored operating system, ie, when the flag element the value "ONE" has a function of the marker element, decompressed by a stored in nonvolatile memory uncompressed routine and is stored in the nonvolatile memory. Before the decompression of the operating system a CRC checksum is calculated and compared with the likewise in the header 28 stored in the storage area 30 by means of a checksum stored in the memory area 32 of the first header 28 length information. Is the calculated checksum matches the checksum stored, the process continues with the decompression. Here, the operating system is decompressed without the header 28 and stored decompressed to the stored in the memory area 36 of the header 28 destination address in the volatile memory. Does the marker element the value "zero" on, is the operating system directly, ie without decompression and without header in the volatile memory copied. After decompress or copying of the operating system as previously described a CRC checksum is determined and compared with a in a second header 40 in volatile memory which is present in decompressed form. If the two checksums match, a startup routine of the operating system is started, whose address is also stored in the header 40 in the storage area 46th

Vers, 3-4

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Vers, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Vers discloses this limitation:

See

> The method involves placing system software into a non-volatile memory (16) in compressed form. After a reset, the software is copied into the volatile memory (18) and is started there. This is done either after decompression or without decompression, depending on the characteristic element. Data corresponding to changes in the user software prior to voltage drop-out are compressed and stored in the non-volatile memory and after reset are decompressed for regeneration of the data in volatile memory.

Vers, Abstract

> The invention is based on the problem to provide a method for operating a data processing device in such a way that non-volatile at least consistent performance storage elements with a minimum memory capacity can be used. the problem is inventively solved in that the system software stored in compressed form in the non-volatile memory and to reset in response to a flag element either is decompressed and stored in the volatile memory or copied without decompression in the volatile memory and started there and / or that the application software associated, are compressed changes of application software files containing or by instructing from the programming device before power failure and stored in the nonvolatile memory and decompressed when power is restored to regenerate the files in the volatile memories.

Vers, 2

> Preferably, a destination address in the volatile memory is in the header in the non-volatile memory, where the system software such as operating system copied or stored decompressed. It is also provided that after decompression or copying of the system software calculated CRC

checksum compared with a in a second header in the region of the beginning of the file system software in the volatile memory, and if they match a start routine is started.

Vers, 2

A particularly preferred compressing algorithm is the LZW algorithm used.

Vers, 2

In Fig. 1 the basic construction of a programmable controller is shown with a bus 12 to which a microprocessor 14, non-volatile and volatile memories 16, 18 and one or more peripheral devices 20 are connected. as non-volatile memory 16 preferably flash memory modules are used, which can be written and read, and in contrast to the volatile memory 18 may be retained their content when they are not supplied with operating voltage. Other non-volatile memory such as EPROM and EEPROM can also be used. Preferably as a volatile memory SRAM and / or DRAM's are used. In Fig. 2, the memory map of the nonvolatile memory 16 is shown. According to the invention, that the system software as the operating system stored in an initial region 22nd Following the operating system is followed by a free space 24, which is a storage area 26 connects with a reset initialization and a decompression algorithm as executable code. Especially with programmable logic controllers or PLCs can also use the software in the nonvolatile memory is stored in compressed form be.

Vers, 3

By a further storage area 36, the destination address is specified in the volatile memory to which the operating system copies or store decompressed. Finally, the tray of the operating system begins in compressed or transparent form.

Vers, 3

To inventively to guarantee the use of non-volatile memories of a minimum memory size is provided that the stored in compressed form in the non-volatile memory operating system during initialization in the volatile memory is loaded. For this purpose it is provided that the non-volatile memory compresses stored operating system, ie, when the flag element the value "ONE" has a function of the marker element, decompressed by a stored in nonvolatile memory uncompressed routine and is stored in the nonvolatile memory. Before the decompression of the

operating system a CRC checksum is calculated and compared with the likewise in the header 28 stored in the storage area 30 by means of a checksum stored in the memory area 32 of the first header 28 length information. Is the calculated checksum matches the checksum stored, the process continues with the decompression. Here, the operating system is decompressed without the header 28 and stored decompressed to the stored in the memory area 36 of the header 28 destination address in the volatile memory. Does the marker element the value "zero" on, is the operating system directly, ie without decompression and without header in the volatile memory copied. After decompress or copying of the operating system as previously described a CRC checksum is determined and compared with a in a second header 40 in volatile memory which is present in decompressed form. If the two checksums match, a startup routine of the operating system is started, whose address is also stored in the header 40 in the storage area 46th

Vers, 3-4

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Vers, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Vers, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Vers, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Vers discloses this limitation:

See Claims 1.3 and 1.4 above.

See also

> A particularly preferred compressing algorithm is the LZW algorithm used.

Vers, 2

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Vers, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 1.3, 1.4, and 15 above.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Vers, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Vers, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Vers, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Vers, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Vers, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Vers, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Vers, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Vers, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Vers, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Vers, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Vers, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Vers, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Vers, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Vers, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Vers, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Vers, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Vers, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Vers, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 15 and 27 above	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Vers, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Vers discloses this limitation: *See* Claims 16 and 27 above	

Appendix A24
Invalidity of U.S. Patent 7,181,608 based on Zwiegincew

U.S. Patent No. 6,317,818 to Zwiegincew ("Zwiegincew") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

In addition, Apple incorporates by reference, as if set forth fully herein, all arguments related to Zwiegincew in pending inter partes review petitions IPR2016-01737, IPR2016-01738, and IPR2016-01739.

Appendix A24
Invalidity of U.S. Patent 7,181,608 based on Zwiegincew

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Zwiegincew, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Zwiegincew discloses this limitation:

> Hard page fault patterns of an application program module are analyzed in order to determine the pages that will be retrieved from disk storage during a common hard page fault scenario. Copies of, or references to, the determined pages are stored in a scenario file, along with an index referred to as a page sequence. The scenario file may also include a prologue indicating events that lead to a hard page fault scenario and an epilogue that may indicate subsequent hard page fault scenarios. Execution of the application program module is monitored to detect the occurrence of a hard page fault scenario. When a hard page fault scenario is detected, a corresponding scenario file is fetched from disk storage and the determined pages, or copies thereof, are transferred into RAM. The determined pages, or copies thereof, may be placed on a stand-by list in RAM and later soft-faulted into the working set of the application program upon request by the application program module, thereby avoiding a sequence of hard page faults.

Zwiegincew, Abstract

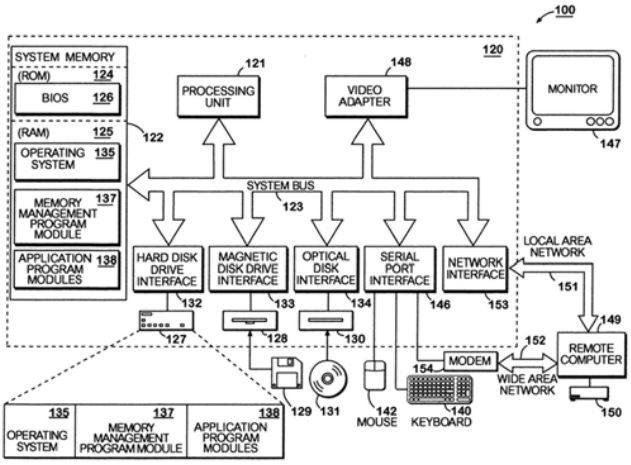


FIG.1

Zwiegincew, Fig. 1

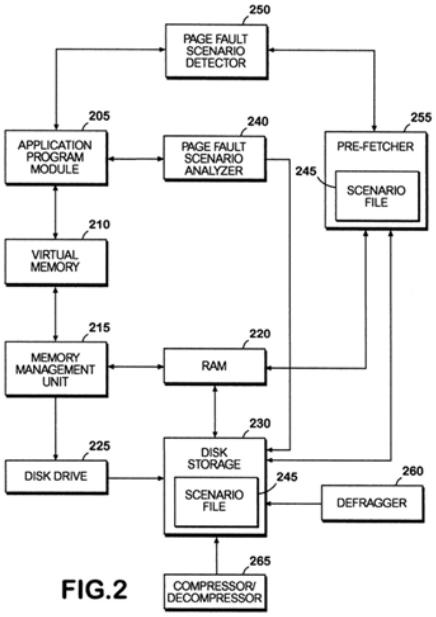


FIG.2

Zwiegincew. Fig 2

There are various potential solutions to the performance bottleneck caused by disk access time during hard page fault scenarios. An obvious potential solution is to reduce disk access time. The reduction of disk access time is primarily a hardware consideration and is not easily

> accomplished. However, other potential solutions involve the manipulation of memory storage through software program modules.

Zwiegincew, 1:45-51

> In an exemplary embodiment, the present invention may comprise one or more memory management program modules 137 stored on the drives or RAM 125 of the computer 100. Specifically, program modules 137 of the present invention may comprise computer implemented instructions for determining which pages will have to be retrieved from disk during a potential hard page fault scenario and pre-fetching the determined pages into RAM prior to the occurrence of the potential hard page fault sequence.

Zwiegincew, 5:50-51.

1.1 maintaining a list of boot data used for booting a computer system;	Zwiegincew, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Zwiegincew discloses this limitation:

> Hard page fault patterns of an application program module are analyzed in order to determine the pages that will be retrieved from disk storage during a common hard page fault scenario. Copies of, or references to, the determined pages are stored in a scenario file, along with an index referred to as a page sequence. The scenario file may also include a prologue indicating events that lead to a hard page fault scenario and an epilogue that may indicate subsequent hard page fault scenarios. Execution of the application program module is monitored to detect the occurrence of a hard page fault scenario. When a hard page fault scenario is detected, a corresponding scenario file is fetched from disk storage and the determined pages, or copies thereof, are transferred into RAM. The determined pages, or copies thereof, may be placed on a stand-by list in RAM and later soft-faulted into the working set of the application program upon request by the application program module, thereby avoiding a sequence of hard page faults.

Zwiegincew, Abstract

> There are various potential solutions to the performance bottleneck caused by disk access time during hard page fault scenarios. An obvious potential solution is to reduce disk access time. The reduction of disk access time is primarily a hardware consideration and is not easily accomplished. However, other potential solutions involve the manipulation of memory storage through software program modules.

Zwiegincew, 1:45-51

> In an exemplary embodiment, the present invention may comprise one or more memory management program modules 137 stored on the drives or RAM 125 of the computer 100. Specifically, program modules 137 of the present invention may comprise computer implemented

> instructions for determining which pages will have to be retrieved from disk during a potential hard page fault scenario and pre-fetching the determined pages into RAM prior to the occurrence of the potential hard page fault sequence.

Zwiegincew, 5:50-51.

> The present invention meets the needs described above by providing a system and method for improving the performance of an application program module by reducing the occurrence of hard page faults during the operation of an application program module. The present invention may be embodied in an add-on software program module that operates in conjunction with the application program module. In this manner, no effort is required on the part of the application programmer to manipulate or modify the application program module in order to improve performance. Furthermore, the add-on software program module does not detract from the intended operation of the application program module.
>
> According to one aspect of the present invention, a scenario file is created which comprises ordered copies of pages that are likely to be retrieved from disk storage by an application program module during a hard page fault. The scenario file is stored in the disk storage. Then, the execution of the application program module is monitored until either an explicit begin-of-scenario instruction is detected, or a hard page fault scenario is detected. A hard page fault scenario may comprise any situation or event that is likely to trigger a hard page fault, i.e., one or more requested pages will not be available in RAM and will be retrieved from disk storage. In response to the detection of a begin-of-scenario instruction or a hard page fault scenario, the scenario file is fetched from disk storage and the ordered copies of the pages are transferred into a standby list in RAM. In this manner, the requested pages will be soft faulted into a working set of the application program module, and no hard page fault will occur. In another aspect of the invention, a hard page fault scenario analyzer is provided for analyzing a hard page fault scenario of an application program module in order to determine the pages that will be retrieved from disk storage upon the occurrence of a hard page fault scenario. The hard page fault scenario analyzer creates a scenario file comprising copies of the pages in the determined order. A hard page fault scenario detector is provided for monitoring the execution of the application module, detecting a hard page fault scenario and sending a message to a pre-fetcher. The hard page fault scenario detector may be manual or automatic. A pre-fetcher retrieves a scenario file from disk storage and transfers the copies of the determined pages into RAM. The copies of the determined pages are

placed on a standby list in RAM. Accordingly, the determined pages will be available in RAM during a hard page fault scenario and will be soft-faulted into the working set of the application program module when they are requested by the application program module, thereby avoiding a hard page fault.

According to another aspect of the invention, a scenario file is created which comprises ordered references to pages that are likely to be retrieved from disk storage by an application program module during a hard page fault scenario, rather than the actual pages themselves. The scenario file is stored in the disk storage. Then, the execution of the application program module is monitored until either an explicit begin-of-scenario instruction is detected, or a hard page fault scenario is detected. A hard page fault scenario may comprise any situation or event that is likely to trigger a hard page fault, i.e., one or more requested pages will not be available in RAM and will be retrieved from disk storage. In response to the detection of a begin-of-scenario instruction or a hard page fault scenario, the pages referenced by the scenario file are fetched from disk storage in the optimal manner and are transferred into a standby list in RAM. In this manner, the requested pages will be soft faulted into a working set of the application program module, and no hard page fault will occur. This aspect of the invention will result in more seek operations on disk, but will still allow reading of the required pages in an optimal manner, rather than the 'as needed' ordering if the pages are hard faulted into RAM. This aspect of the invention also reduces the disk space requirements over the previously mentioned aspect.

Zwiegincew, 2:43-3:49.

A hard page fault scenario analyzer 240 anticipates and analyzes hard page fault scenarios. As mentioned, a hard page fault scenario is a situation in which a hard page fault sequence is highly likely to occur. The hard page fault scenario analyzer logs various hard page fault scenarios that occur during operation of the application program module 205. The logged hard page fault scenarios are then analyzed to determine if they re-occur frequently, and if they do, they are put in a scenario file. This analysis can occur programmatically on the end-user's computer system, or in advance by the developer of a particular software product. As an example, suppose the application program module 205 is a word processor and that an anticipated hard page fault scenario is the situation in which the user selects a well known "file open" command. In response to the "file open" command, the application program will display a graphical representation of a file directory. However, in order to display the graphical representation of

> the file directory, a sequence of hard page faults will occur because the word processor must retrieve a particular set of pages from disk. In accordance with an exemplary embodiment of the present invention, the hard page fault scenario analyzer 240 anticipates the "open file" hard page fault scenario of the example and determines the set of pages that will need to be retrieved from disk upon the occurrence of the hard page fault. The determination of pages that will need to be retrieved from disk will be described in greater detail below. The detection of particular classes of hard page fault scenarios may be built into the system. For example, application launch is virtually always a hard page fault scenario, so an exemplary embodiment of the present invention may be configured such that any launch operation of an application program will be considered to be a hard page fault scenario.

Zwiegincew, 6:29-61.

> The hard page fault scenario analyzer 240 may comprise functionality for automatically analyzing hard page fault scenarios and generating corresponding scenario files. By way of illustration, the hard page fault analyzer 240 may log hard page faults that occur upon execution of a process during operation of an application program module 205. During idle time of the application program module 205, the hard page fault scenario analyzer 240 may write the log of hard page faults to a log file. Then, a pattern matching algorithm may be used to find a pattern of hard page faults based on all log files generated for the process same. If a pattern of hard page faults is found, a new scenario file may be generated based on the pages that are retrieved from disk during the pattern. Automatically generated scenario files may be subject to subsequent refinement, i.e., they may be input into the pattern-matching algorithm.

Zwiegincew, 7:24-40

See also Zwiegincew. 1:5-2:40, Figs 1-2

1.2 initializing a central processing unit of the computer system;	Zwiegincew, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Zwiegincew, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Zwiegincew discloses this limitation:

> Hard page fault patterns of an application program module are analyzed in order to determine the pages that will be retrieved from disk storage during a common hard page fault scenario. Copies of, or references to, the determined pages are stored in a scenario file, along with an index referred to as a page sequence. The scenario file may also include a prologue indicating events that lead to a hard page fault scenario and an epilogue that may indicate subsequent hard page fault scenarios. Execution of the application program module is monitored to detect the occurrence of a hard page fault scenario. When a hard page fault scenario is detected, a corresponding scenario file is fetched from disk storage and the determined pages, or copies thereof, are transferred into RAM. The determined pages, or copies thereof, may be placed on a stand-by list in RAM and later soft-faulted into the working set of the application program upon request by the application program module, thereby avoiding a sequence of hard page faults.

Zwiegincew, Abstract

> There are various potential solutions to the performance bottleneck caused by disk access time during hard page fault scenarios. An obvious potential solution is to reduce disk access time. The reduction of disk access time is primarily a hardware consideration and is not easily

> accomplished. However, other potential solutions involve the manipulation of memory storage through software program modules.

Zwiegincew, 1:45-51

> In an exemplary embodiment, the present invention may comprise one or more memory management program modules 137 stored on the drives or RAM 125 of the computer 100. Specifically, program modules 137 of the present invention may comprise computer implemented instructions for determining which pages will have to be retrieved from disk during a potential hard page fault scenario and pre-fetching the determined pages into RAM prior to the occurrence of the potential hard page fault sequence.

Zwiegincew, 5:50-51.

> The present invention meets the needs described above by providing a system and method for improving the performance of an application program module by reducing the occurrence of hard page faults during the operation of an application program module. The present invention may be embodied in an add-on software program module that operates in conjunction with the application program module. In this manner, no effort is required on the part of the application programmer to manipulate or modify the application program module in order to improve performance. Furthermore, the add-on software program module does not detract from the intended operation of the application program module.
>
> According to one aspect of the present invention, a scenario file is created which comprises ordered copies of pages that are likely to be retrieved from disk storage by an application program module during a hard page fault. The scenario file is stored in the disk storage. Then, the execution of the application program module is monitored until either an explicit begin-of-scenario instruction is detected, or a hard page fault scenario is detected. A hard page fault scenario may comprise any situation or event that is likely to trigger a hard page fault, i.e., one or more requested pages will not be available in RAM and will be retrieved from disk storage. In response to the detection of a begin-of-scenario instruction or a hard page fault scenario, the scenario file is fetched from disk storage and the ordered copies of the pages are transferred into a standby list in RAM. In this manner, the requested pages will be soft faulted into a working set of the application program module, and no hard page fault will occur. In another aspect of the invention, a hard page fault scenario analyzer is provided for analyzing a hard page fault scenario of an application program module in order to

> determine the pages that will be retrieved from disk storage upon the occurrence of a hard page fault scenario. The hard page fault scenario analyzer creates a scenario file comprising copies of the pages in the determined order. A hard page fault scenario detector is provided for monitoring the execution of the application module, detecting a hard page fault scenario and sending a message to a pre-fetcher. The hard page fault scenario detector may be manual or automatic. A pre-fetcher retrieves a scenario file from disk storage and transfers the copies of the determined pages into RAM. The copies of the determined pages are placed on a standby list in RAM. Accordingly, the determined pages will be available in RAM during a hard page fault scenario and will be soft-faulted into the working set of the application program module when they are requested by the application program module, thereby avoiding a hard page fault.
>
> According to another aspect of the invention, a scenario file is created which comprises ordered references to pages that are likely to be retrieved from disk storage by an application program module during a hard page fault scenario, rather than the actual pages themselves. The scenario file is stored in the disk storage. Then, the execution of the application program module is monitored until either an explicit begin-of-scenario instruction is detected, or a hard page fault scenario is detected. A hard page fault scenario may comprise any situation or event that is likely to trigger a hard page fault, i.e., one or more requested pages will not be available in RAM and will be retrieved from disk storage. In response to the detection of a begin-of-scenario instruction or a hard page fault scenario, the pages referenced by the scenario file are fetched from disk storage in the optimal manner and are transferred into a standby list in RAM. In this manner, the requested pages will be soft faulted into a working set of the application program module, and no hard page fault will occur. This aspect of the invention will result in more seek operations on disk, but will still allow reading of the required pages in an optimal manner, rather than the 'as needed' ordering if the pages are hard faulted into RAM. This aspect of the invention also reduces the disk space requirements over the previously mentioned aspect.

Zwiegincew, 2:43-3:49.

See also Zwiegincew. 1:5-2:40, Figs 1-2

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Zwiegincew, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Zwiegincew discloses this limitation:

> Hard page fault patterns of an application program module are analyzed in order to determine the pages that will be retrieved from disk storage during a common hard page fault scenario. Copies of, or references to, the determined pages are stored in a scenario file, along with an index referred to as a page sequence. The scenario file may also include a prologue indicating events that lead to a hard page fault scenario and an epilogue that may indicate subsequent hard page fault scenarios. Execution of the application program module is monitored to detect the occurrence of a hard page fault scenario. When a hard page fault scenario is detected, a corresponding scenario file is fetched from disk storage and the determined pages, or copies thereof, are transferred into RAM. The determined pages, or copies thereof, may be placed on a stand-by list in RAM and later soft-faulted into the working set of the application program upon request by the application program module, thereby avoiding a sequence of hard page faults.

Zwiegincew, Abstract

> There are various potential solutions to the performance bottleneck caused by disk access time during hard page fault scenarios. An obvious potential solution is to reduce disk access time. The reduction of disk access time is primarily a hardware consideration and is not easily accomplished. However, other potential solutions involve the manipulation of memory storage through software program modules.

Zwiegincew, 1:45-51

> In an exemplary embodiment, the present invention may comprise one or more memory management program modules 137 stored on the drives or RAM 125 of the computer 100. Specifically, program modules 137 of the present invention may comprise computer implemented instructions for determining which pages will have to be retrieved from disk during a potential hard page fault scenario and pre-fetching the determined pages into RAM prior to the occurrence of the potential hard page fault sequence.

Zwiegincew, 5:50-51.

> The present invention meets the needs described above by providing a system and method for improving the performance of an application program module by reducing the occurrence of hard page faults during the operation of an application program module. The present invention may be embodied in an add-on software program module that operates in conjunction with the application program module. In this manner, no effort is required on the part of the application programmer to manipulate or modify the application program module in order to improve performance. Furthermore, the add-on software program module does not detract from the intended operation of the application program module.
>
> According to one aspect of the present invention, a scenario file is created which comprises ordered copies of pages that are likely to be retrieved from disk storage by an application program module during a hard page fault. The scenario file is stored in the disk storage. Then, the execution of the application program module is monitored until either an explicit begin-of-scenario instruction is detected, or a hard page fault scenario is detected. A hard page fault scenario may comprise any situation or event that is likely to trigger a hard page fault, i.e., one or more requested pages will not be available in RAM and will be retrieved from disk storage. In response to the detection of a begin-of-scenario instruction or a hard page fault scenario, the scenario file is fetched from disk storage and the ordered copies of the pages are

transferred into a standby list in RAM. In this manner, the requested pages will be soft faulted into a working set of the application program module, and no hard page fault will occur. In another aspect of the invention, a hard page fault scenario analyzer is provided for analyzing a hard page fault scenario of an application program module in order to determine the pages that will be retrieved from disk storage upon the occurrence of a hard page fault scenario. The hard page fault scenario analyzer creates a scenario file comprising copies of the pages in the determined order. A hard page fault scenario detector is provided for monitoring the execution of the application module, detecting a hard page fault scenario and sending a message to a pre-fetcher. The hard page fault scenario detector may be manual or automatic. A pre-fetcher retrieves a scenario file from disk storage and transfers the copies of the determined pages into RAM. The copies of the determined pages are placed on a standby list in RAM. Accordingly, the determined pages will be available in RAM during a hard page fault scenario and will be soft-faulted into the working set of the application program module when they are requested by the application program module, thereby avoiding a hard page fault.

According to another aspect of the invention, a scenario file is created which comprises ordered references to pages that are likely to be retrieved from disk storage by an application program module during a hard page fault scenario, rather than the actual pages themselves. The scenario file is stored in the disk storage. Then, the execution of the application program module is monitored until either an explicit begin-of-scenario instruction is detected, or a hard page fault scenario is detected. A hard page fault scenario may comprise any situation or event that is likely to trigger a hard page fault, i.e., one or more requested pages will not be available in RAM and will be retrieved from disk storage. In response to the detection of a begin-of-scenario instruction or a hard page fault scenario, the pages referenced by the scenario file are fetched from disk storage in the optimal manner and are transferred into a standby list in RAM. In this manner, the requested pages will be soft faulted into a working set of the application program module, and no hard page fault will occur. This aspect of the invention will result in more seek operations on disk, but will still allow reading of the required pages in an optimal manner, rather than the 'as needed' ordering if the pages are hard faulted into RAM. This aspect of the invention also reduces the disk space requirements over the previously mentioned aspect.

Zwiegincew, 2:43-3:49.

> Further, an exemplary embodiment includes a disk compressor/decompressor 265. Well known compression algorithms may be employed to achieve approximately 50% compression with 25 MB/s decompression throughput. These results may be achieved with as little as 64 KB extra memory. Average disk transfer rates are about 8 MB/s. So, for an illustrative 3 MB pre-fetch scenario, comparative pre-fetch times are as follows:
>
> No compression: 0.012 s (seek)+3 MB/8 MB/s (read)=0.3870 s.
>
> 50% compression: 0.012 s (seek)+1.5 MB/8 MB/s (read)+3 MB/25 MB/s (decompress)=0.3195 s.
>
> Thus, there is a 17.5% improvement in pre-fetch time using 50% compression.

Zwiegincew, 8:66-9:13

See also Zwiegincew. 1:5-2:40, Figs 1-2

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Zwiegincew, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Zwiegincew, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.3, and 1.4 above.	

4. The method of claim 1, further comprising updating the list of boot data.	Zwiegincew, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Zwiegincew discloses this limitation:

See Claim 1.1 above.

See also

> Hard page fault patterns of an application program module are analyzed in order to determine the pages that will be retrieved from disk storage during a common hard page fault scenario. Copies of, or references to, the determined pages are stored in a scenario file, along with an index referred to as a page sequence. The scenario file may also include a prologue indicating events that lead to a hard page fault scenario and an epilogue that may indicate subsequent hard page fault scenarios. Execution of the application program module is monitored to detect the occurrence of a hard page fault scenario. When a hard page fault scenario is detected, a corresponding scenario file is fetched from disk storage and the determined pages, or copies thereof, are transferred into RAM. The determined pages, or copies thereof, may be placed on a stand-by list in RAM and later soft-faulted into the working set of the application program upon request by the application program module, thereby avoiding a sequence of hard page faults.

Zwiegincew, Abstract

> The present invention meets the needs described above by providing a system and method for improving the performance of an application program module by reducing the occurrence of hard page faults during the operation of an application program module. The present invention may be embodied in an add-on software program module that operates in conjunction with the application program module. In this manner, no effort is required on the part of the application programmer to manipulate or modify the application program module in order to improve performance. Furthermore, the add-on software program module does not detract from the intended operation of the application program module.

According to one aspect of the present invention, a scenario file is created which comprises ordered copies of pages that are likely to be retrieved from disk storage by an application program module during a hard page fault. The scenario file is stored in the disk storage. Then, the execution of the application program module is monitored until either an explicit begin-of-scenario instruction is detected, or a hard page fault scenario is detected. A hard page fault scenario may comprise any situation or event that is likely to trigger a hard page fault, i.e., one or more requested pages will not be available in RAM and will be retrieved from disk storage. In response to the detection of a begin-of-scenario instruction or a hard page fault scenario, the scenario file is fetched from disk storage and the ordered copies of the pages are transferred into a standby list in RAM. In this manner, the requested pages will be soft faulted into a working set of the application program module, and no hard page fault will occur. In another aspect of the invention, a hard page fault scenario analyzer is provided for analyzing a hard page fault scenario of an application program module in order to determine the pages that will be retrieved from disk storage upon the occurrence of a hard page fault scenario. The hard page fault scenario analyzer creates a scenario file comprising copies of the pages in the determined order. A hard page fault scenario detector is provided for monitoring the execution of the application module, detecting a hard page fault scenario and sending a message to a pre-fetcher. The hard page fault scenario detector may be manual or automatic. A pre-fetcher retrieves a scenario file from disk storage and transfers the copies of the determined pages into RAM. The copies of the determined pages are placed on a standby list in RAM. Accordingly, the determined pages will be available in RAM during a hard page fault scenario and will be soft-faulted into the working set of the application program module when they are requested by the application program module, thereby avoiding a hard page fault.

According to another aspect of the invention, a scenario file is created which comprises ordered references to pages that are likely to be retrieved from disk storage by an application program module during a hard page fault scenario, rather than the actual pages themselves. The scenario file is stored in the disk storage. Then, the execution of the application program module is monitored until either an explicit begin-of-scenario instruction is detected, or a hard page fault scenario is detected. A hard page fault scenario may comprise any situation or event that is likely to trigger a hard page fault, i.e., one or more requested pages will not be available in RAM and will be retrieved from disk storage. In response to the detection of a begin-of-scenario instruction or a hard page fault scenario, the pages referenced by the

> scenario file are fetched from disk storage in the optimal manner and are transferred into a standby list in RAM. In this manner, the requested pages will be soft faulted into a working set of the application program module, and no hard page fault will occur. This aspect of the invention will result in more seek operations on disk, but will still allow reading of the required pages in an optimal manner, rather than the 'as needed' ordering if the pages are hard faulted into RAM. This aspect of the invention also reduces the disk space requirements over the previously mentioned aspect.

Zwiegincew, 2:43-3:49.

> A hard page fault scenario analyzer 240 anticipates and analyzes hard page fault scenarios. As mentioned, a hard page fault scenario is a situation in which a hard page fault sequence is highly likely to occur. The hard page fault scenario analyzer logs various hard page fault scenarios that occur during operation of the application program module 205. The logged hard page fault scenarios are then analyzed to determine if they re-occur frequently, and if they do, they are put in a scenario file. This analysis can occur programmatically on the end-user's computer system, or in advance by the developer of a particular software product. As an example, suppose the application program module 205 is a word processor and that an anticipated hard page fault scenario is the situation in which the user selects a well known "file open" command. In response to the "file open" command, the application program will display a graphical representation of a file directory. However, in order to display the graphical representation of the file directory, a sequence of hard page faults will occur because the word processor must retrieve a particular set of pages from disk. In accordance with an exemplary embodiment of the present invention, the hard page fault scenario analyzer 240 anticipates the "open file" hard page fault scenario of the example and determines the set of pages that will need to be retrieved from disk upon the occurrence of the hard page fault. The determination of pages that will need to be retrieved from disk will be described in greater detail below. The detection of particular classes of hard page fault scenarios may be built into the system. For example, application launch is virtually always a hard page fault scenario, so an exemplary embodiment of the present invention may be configured such that any launch operation of an application program will be considered to be a hard page fault scenario.

Zwiegincew, 6:29-61.

> The hard page fault scenario analyzer 240 may comprise functionality for automatically analyzing hard page fault scenarios and generating

> corresponding scenario files. By way of illustration, the hard page fault analyzer 240 may log hard page faults that occur upon execution of a process during operation of an application program module 205. During idle time of the application program module 205, the hard page fault scenario analyzer 240 may write the log of hard page faults to a log file. Then, a pattern matching algorithm may be used to find a pattern of hard page faults based on all log files generated for the process same. If a pattern of hard page faults is found, a new scenario file may be generated based on the pages that are retrieved from disk during the pattern. Automatically generated scenario files may be subject to subsequent refinement, i.e., they may be input into the pattern-matching algorithm.

Zwiegincew, 7:24-40

> Further, an exemplary embodiment includes a disk compressor/decompressor 265. Well known compression algorithms may be employed to achieve approximately 50% compression with 25 MB/s decompression throughput. These results may be achieved with as little as 64 KB extra memory. Average disk transfer rates are about 8 MB/s. So, for an illustrative 3 MB pre-fetch scenario, comparative pre-fetch times are as follows:
>
> No compression: 0.012 s (seek)+3 MB/8 MB/s (read)=0.3870 s.
>
> 50% compression: 0.012 s (seek)+1.5 MB/8 MB/s (read)+3 MB/25 MB/s (decompress)=0.3195 s.
>
> Thus, there is a 17.5% improvement in pre-fetch time using 50% compression.

Zwiegincew, 8:66-9:13

See also Zwiegincew. 1:5-2:40, 5:50-51, Figs 1-2

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Zwiegincew, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Zwiegincew, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Zwiegincew, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Zwiegincew, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

7.2 a cache memory device; and;	Zwiegincew, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Zwiegincew, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Zwiegincew, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Zwiegincew, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Zwiegincew, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.	

9.1 maintaining a list of application data associated with an application program;	Zwiegincew, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Zwiegincew, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Zwiegincew, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.4, 2, and 8 above.	

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Zwiegincew, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Zwiegincew discloses this limitation:

> Hard page fault patterns of an application program module are analyzed in order to determine the pages that will be retrieved from disk storage during a common hard page fault scenario. Copies of, or references to, the determined pages are stored in a scenario file, along with an index referred to as a page sequence. The scenario file may also include a prologue indicating events that lead to a hard page fault scenario and an epilogue that may indicate subsequent hard page fault scenarios. Execution of the application program module is monitored to detect the occurrence of a hard page fault scenario. When a hard page fault scenario is detected, a corresponding scenario file is fetched from disk storage and the determined pages, or copies thereof, are transferred into RAM. The determined pages, or copies thereof, may be placed on a stand-by list in RAM and later soft-faulted into the working set of the application program upon request by the application program module, thereby avoiding a sequence of hard page faults.

Zwiegincew, Abstract

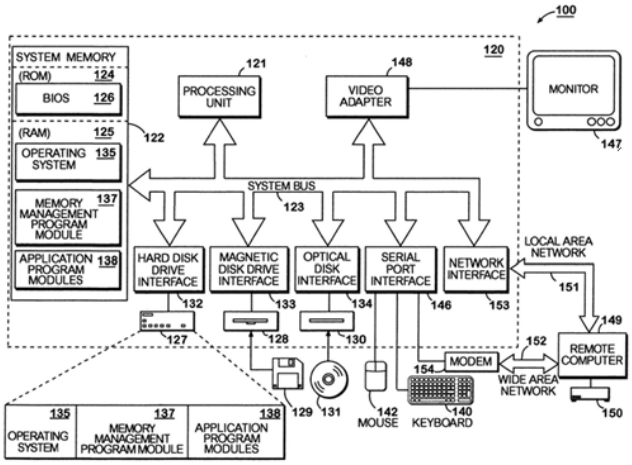


FIG.1

Zwiegincew, Fig. 1

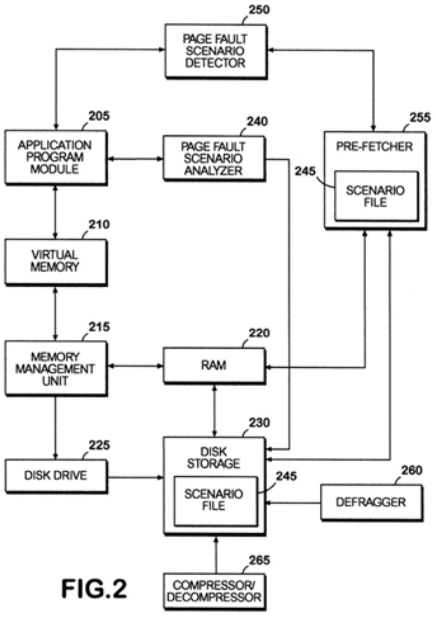


FIG.2

Zwiegincew. Fig 2

There are various potential solutions to the performance bottleneck caused by disk access time during hard page fault scenarios. An obvious potential solution is to reduce disk access time. The reduction of disk access time is primarily a hardware consideration and is not easily

> accomplished. However, other potential solutions involve the manipulation of memory storage through software program modules.

Zwiegincew, 1:45-51

> In an exemplary embodiment, the present invention may comprise one or more memory management program modules 137 stored on the drives or RAM 125 of the computer 100. Specifically, program modules 137 of the present invention may comprise computer implemented instructions for determining which pages will have to be retrieved from disk during a potential hard page fault scenario and pre-fetching the determined pages into RAM prior to the occurrence of the potential hard page fault sequence.

Zwiegincew, 5:50-51.

> The present invention meets the needs described above by providing a system and method for improving the performance of an application program module by reducing the occurrence of hard page faults during the operation of an application program module. The present invention may be embodied in an add-on software program module that operates in conjunction with the application program module. In this manner, no effort is required on the part of the application programmer to manipulate or modify the application program module in order to improve performance. Furthermore, the add-on software program module does not detract from the intended operation of the application program module.
>
> According to one aspect of the present invention, a scenario file is created which comprises ordered copies of pages that are likely to be retrieved from disk storage by an application program module during a hard page fault. The scenario file is stored in the disk storage. Then, the execution of the application program module is monitored until either an explicit begin-of-scenario instruction is detected, or a hard page fault scenario is detected. A hard page fault scenario may comprise any situation or event that is likely to trigger a hard page fault, i.e., one or more requested pages will not be available in RAM and will be retrieved from disk storage. In response to the detection of a begin-of-scenario instruction or a hard page fault scenario, the scenario file is fetched from disk storage and the ordered copies of the pages are transferred into a standby list in RAM. In this manner, the requested pages will be soft faulted into a working set of the application program module, and no hard page fault will occur. In another aspect of the invention, a hard page fault scenario analyzer is provided for analyzing a hard page fault scenario of an application program module in order to

> determine the pages that will be retrieved from disk storage upon the occurrence of a hard page fault scenario. The hard page fault scenario analyzer creates a scenario file comprising copies of the pages in the determined order. A hard page fault scenario detector is provided for monitoring the execution of the application module, detecting a hard page fault scenario and sending a message to a pre-fetcher. The hard page fault scenario detector may be manual or automatic. A pre-fetcher retrieves a scenario file from disk storage and transfers the copies of the determined pages into RAM. The copies of the determined pages are placed on a standby list in RAM. Accordingly, the determined pages will be available in RAM during a hard page fault scenario and will be soft-faulted into the working set of the application program module when they are requested by the application program module, thereby avoiding a hard page fault.
>
> According to another aspect of the invention, a scenario file is created which comprises ordered references to pages that are likely to be retrieved from disk storage by an application program module during a hard page fault scenario, rather than the actual pages themselves. The scenario file is stored in the disk storage. Then, the execution of the application program module is monitored until either an explicit begin-of-scenario instruction is detected, or a hard page fault scenario is detected. A hard page fault scenario may comprise any situation or event that is likely to trigger a hard page fault, i.e., one or more requested pages will not be available in RAM and will be retrieved from disk storage. In response to the detection of a begin-of-scenario instruction or a hard page fault scenario, the pages referenced by the scenario file are fetched from disk storage in the optimal manner and are transferred into a standby list in RAM. In this manner, the requested pages will be soft faulted into a working set of the application program module, and no hard page fault will occur. This aspect of the invention will result in more seek operations on disk, but will still allow reading of the required pages in an optimal manner, rather than the 'as needed' ordering if the pages are hard faulted into RAM. This aspect of the invention also reduces the disk space requirements over the previously mentioned aspect.

Zwiegincew, 2:43-3:49.

> A hard page fault scenario analyzer 240 anticipates and analyzes hard page fault scenarios. As mentioned, a hard page fault scenario is a situation in which a hard page fault sequence is highly likely to occur. The hard page fault scenario analyzer logs various hard page fault scenarios that occur during operation of the application program module 205. The logged hard page fault scenarios are then analyzed to

> determine if they re-occur frequently, and if they do, they are put in a scenario file. This analysis can occur programmatically on the end-user's computer system, or in advance by the developer of a particular software product. As an example, suppose the application program module 205 is a word processor and that an anticipated hard page fault scenario is the situation in which the user selects a well known "file open" command. In response to the "file open" command, the application program will display a graphical representation of a file directory. However, in order to display the graphical representation of the file directory, a sequence of hard page faults will occur because the word processor must retrieve a particular set of pages from disk. In accordance with an exemplary embodiment of the present invention, the hard page fault scenario analyzer 240 anticipates the "open file" hard page fault scenario of the example and determines the set of pages that will need to be retrieved from disk upon the occurrence of the hard page fault. The determination of pages that will need to be retrieved from disk will be described in greater detail below. The detection of particular classes of hard page fault scenarios may be built into the system. For example, application launch is virtually always a hard page fault scenario, so an exemplary embodiment of the present invention may be configured such that any launch operation of an application program will be considered to be a hard page fault scenario.

Zwiegincew, 6:29-61.

> The hard page fault scenario analyzer 240 may comprise functionality for automatically analyzing hard page fault scenarios and generating corresponding scenario files. By way of illustration, the hard page fault analyzer 240 may log hard page faults that occur upon execution of a process during operation of an application program module 205. During idle time of the application program module 205, the hard page fault scenario analyzer 240 may write the log of hard page faults to a log file. Then, a pattern matching algorithm may be used to find a pattern of hard page faults based on all log files generated for the process same. If a pattern of hard page faults is found, a new scenario file may be generated based on the pages that are retrieved from disk during the pattern. Automatically generated scenario files may be subject to subsequent refinement, i.e., they may be input into the pattern-matching algorithm.

Zwiegincew, 7:24-40

See also Zwiegincew. 1:5-2:40, Figs 1-2

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Zwiegincew, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Zwiegincew discloses this limitation:

> Hard page fault patterns of an application program module are analyzed in order to determine the pages that will be retrieved from disk storage during a common hard page fault scenario. Copies of, or references to, the determined pages are stored in a scenario file, along with an index referred to as a page sequence. The scenario file may also include a prologue indicating events that lead to a hard page fault scenario and an epilogue that may indicate subsequent hard page fault scenarios. Execution of the application program module is monitored to detect the occurrence of a hard page fault scenario. When a hard page fault scenario is detected, a corresponding scenario file is fetched from disk storage and the determined pages, or copies thereof, are transferred into RAM. The determined pages, or copies thereof, may be placed on a stand-by list in RAM and later soft-faulted into the working set of the application program upon request by the application program module, thereby avoiding a sequence of hard page faults.

Zwiegincew, Abstract

> There are various potential solutions to the performance bottleneck caused by disk access time during hard page fault scenarios. An obvious potential solution is to reduce disk access time. The reduction of disk access time is primarily a hardware consideration and is not easily accomplished. However, other potential solutions involve the manipulation of memory storage through software program modules.

Zwiegincew, 1:45-51

> In an exemplary embodiment, the present invention may comprise one or more memory management program modules 137 stored on the drives or RAM 125 of the computer 100. Specifically, program modules 137 of the present invention may comprise computer implemented instructions for determining which pages will have to be retrieved from

> disk during a potential hard page fault scenario and pre-fetching the determined pages into RAM prior to the occurrence of the potential hard page fault sequence.

Zwiegincew, 5:50-51.

> The present invention meets the needs described above by providing a system and method for improving the performance of an application program module by reducing the occurrence of hard page faults during the operation of an application program module. The present invention may be embodied in an add-on software program module that operates in conjunction with the application program module. In this manner, no effort is required on the part of the application programmer to manipulate or modify the application program module in order to improve performance. Furthermore, the add-on software program module does not detract from the intended operation of the application program module.
>
> According to one aspect of the present invention, a scenario file is created which comprises ordered copies of pages that are likely to be retrieved from disk storage by an application program module during a hard page fault. The scenario file is stored in the disk storage. Then, the execution of the application program module is monitored until either an explicit begin-of-scenario instruction is detected, or a hard page fault scenario is detected. A hard page fault scenario may comprise any situation or event that is likely to trigger a hard page fault, i.e., one or more requested pages will not be available in RAM and will be retrieved from disk storage. In response to the detection of a begin-of-scenario instruction or a hard page fault scenario, the scenario file is fetched from disk storage and the ordered copies of the pages are transferred into a standby list in RAM. In this manner, the requested pages will be soft faulted into a working set of the application program module, and no hard page fault will occur. In another aspect of the invention, a hard page fault scenario analyzer is provided for analyzing a hard page fault scenario of an application program module in order to determine the pages that will be retrieved from disk storage upon the occurrence of a hard page fault scenario. The hard page fault scenario analyzer creates a scenario file comprising copies of the pages in the determined order. A hard page fault scenario detector is provided for monitoring the execution of the application module, detecting a hard page fault scenario and sending a message to a pre-fetcher. The hard page fault scenario detector may be manual or automatic. A pre-fetcher retrieves a scenario file from disk storage and transfers the copies of the determined pages into RAM. The copies of the determined pages are placed on a standby list in RAM. Accordingly, the determined pages

> will be available in RAM during a hard page fault scenario and will be soft-faulted into the working set of the application program module when they are requested by the application program module, thereby avoiding a hard page fault.
>
> According to another aspect of the invention, a scenario file is created which comprises ordered references to pages that are likely to be retrieved from disk storage by an application program module during a hard page fault scenario, rather than the actual pages themselves. The scenario file is stored in the disk storage. Then, the execution of the application program module is monitored until either an explicit begin-of-scenario instruction is detected, or a hard page fault scenario is detected. A hard page fault scenario may comprise any situation or event that is likely to trigger a hard page fault, i.e., one or more requested pages will not be available in RAM and will be retrieved from disk storage. In response to the detection of a begin-of-scenario instruction or a hard page fault scenario, the pages referenced by the scenario file are fetched from disk storage in the optimal manner and are transferred into a standby list in RAM. In this manner, the requested pages will be soft faulted into a working set of the application program module, and no hard page fault will occur. This aspect of the invention will result in more seek operations on disk, but will still allow reading of the required pages in an optimal manner, rather than the 'as needed' ordering if the pages are hard faulted into RAM. This aspect of the invention also reduces the disk space requirements over the previously mentioned aspect.

Zwiegincew, 2:43-3:49.

> Further, an exemplary embodiment includes a disk compressor/decompressor 265. Well known compression algorithms may be employed to achieve approximately 50% compression with 25 MB/s decompression throughput. These results may be achieved with as little as 64 KB extra memory. Average disk transfer rates are about 8 MB/s. So, for an illustrative 3 MB pre-fetch scenario, comparative pre-fetch times are as follows:
>
> No compression: 0.012 s (seek)+3 MB/8 MB/s (read)=0.3870 s.
>
> 50% compression: 0.012 s (seek)+1.5 MB/8 MB/s (read)+3 MB/25 MB/s (decompress)=0.3195 s.
>
> Thus, there is a 17.5% improvement in pre-fetch time using 50% compression.

Zwiegincew, 8:66-9:13 *See also* Zwiegincew. 1:5-2:40, Figs 1-2

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Zwiegincew, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Zwiegincew, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Zwiegincew, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Zwiegincew, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.3, 1.4, and 15 above.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Zwiegincew, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Zwiegincew, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Zwiegincew, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Zwiegincew, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Zwiegincew, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Zwiegincew, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Zwiegincew, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Zwiegincew, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Zwiegincew, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Zwiegincew, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Zwiegincew, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

27.3 cache memory; and.	Zwiegincew, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Zwiegincew, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Zwiegincew, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Zwiegincew, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Zwiegincew, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Zwiegincew, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Zwiegincew, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 15 and 27 above	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Zwiegincew, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Zwiegincew discloses this limitation: *See* Claims 16 and 27 above	

Appendix A25
Invalidity of U.S. Patent 7,181,608 based on Anyimi

Anyimi, "Implementing a Plug and Play BIOS Using Intel's Boot Block Flash Memory," Feb. 1995 ("Anyimi") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A25
Invalidity of U.S. Patent 7,181,608 based on Anyimi

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Anyimi, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Anyimi discloses this claim limitation:

Plug and Play can make the usual experience of adding new functionality to an existing system as easy as:

1. Turn the system off.
2. Insert the new device.
3. Turn the system on.

PnP flash BIOS can also extend the life of the PC. By enabling simple updates to the BIOS (simply insert the upgrade disk and the software programs the new BIOS), the user can get more out of their PC investment.

Anyimi, 1.0

3.1 PnP Components

For a system to be fully PnP-compliant, four basic elements are required:

1. System/PnP BIOS
2. PnP operating system
3. PnP hardware devices
4. PnP application software

Anyimi, 3.1

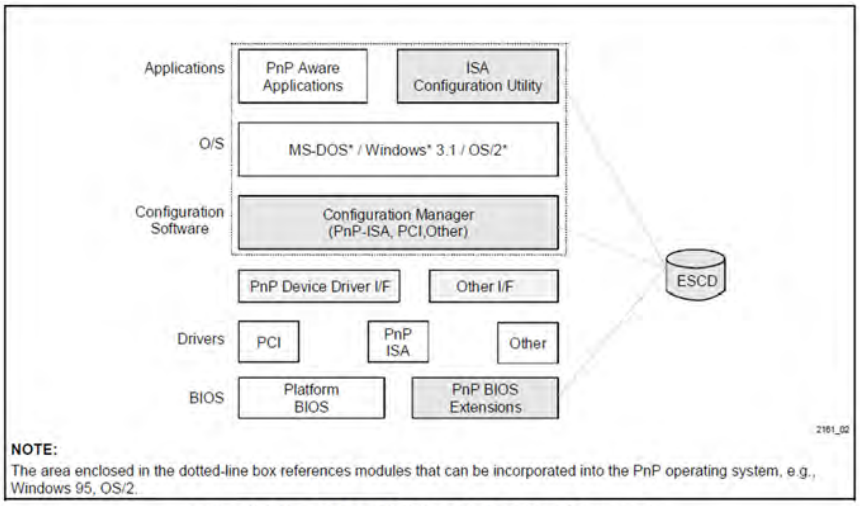


Figure 2. Plug and Play Software Architecture Components.
Note the many layers that depend on the ESCD. The parameter blocks of the boot block flash memory enable this nonvolatile storage capability of Plug and Play.

Anyimi, Fig. 2

Today, PnP BIOS requirements may extend beyond 128 KB. A standard system BIOS consumes roughly 64 KB; PnP support is another 12 KB; automatic power management support adds 2 to 10 KB; PCI (Peripheral Components Interconnect) extensions hoard some 20 KB of the BIOS; and on-board VGA (Video Graphics Adapter) controllers utilize an extra 30 KB to 40 KB of space. The potential size of the BIOS for such a system is almost 150 KB. Although most of this code may be compressed, certain code segments cannot; besides even compression has its limits. Since new developments for the PC show no signs of abatement, it seems clear that a means of updating as well as expanding BIOS quickly and easily has become a necessity. Plug and Play is only one of the many technologies that can be implemented better with flash. As users realize the benefits of PnP, the demand generated will drive the number of Plug and Play systems upward.

Anyimi, 2.2

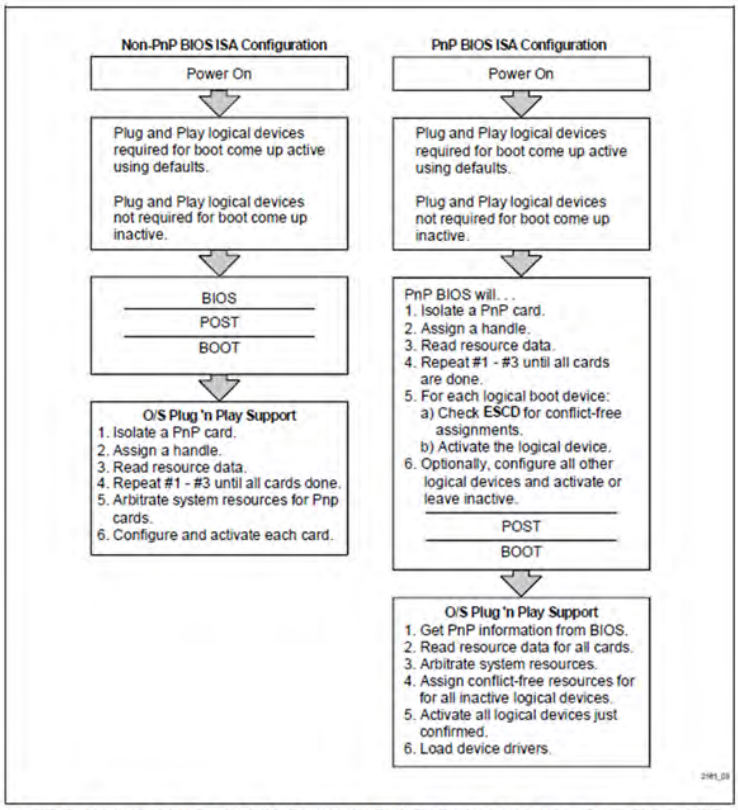


Figure 3. Possible PnP/Non-PnP BIOS ISA Add-In Card Configuration Flow. Note the importance of the ESCD in the Plug and Play Configuration Flow

Anyimi, Fig. 3

Furthermore, the PnP BIOS is capable of event management. Through its event management interfaces, the PnP BIOS can alert the system about new devices, like a notebook docking station, added or removed during runtime.

The POST procedure of the PnP BIOS identifies, tests, and configures the system before passing control to the operating system. The POST process must maintain previous POST compatibility, configure all legacy devices known to the PnP BIOS, arbitrate resources, initialize the IPL (Initial Program Load—this is how the operating system is launched), and support both PnP and non-PnP operating systems. Upon completion of the POST process, the BIOS attempts to have all necessary system devices initialized and enabled before the operating system is loaded; the PnP BIOS aspires further to provide the operating system a conflict-free environment in which to boot.

Anyimi, 3.2

Without an ESCD, the PnP BIOS must perform resource allocation as well as conflict detection and resolution each and every time the system is booted, and any locking of devices must be done by the supporting PnP operating system. Although the need for nonvolatile storage cannot be disputed, the amount of storage required depends on whether or not the PnP system needs real time updates. Ultimately, it will be decided by the methodology selected for resource management. A static or dynamic allocation scheme requires a fixed amount of nonvolatile memory for the ESCD and other BIOS parameters. A combined scheme for allocation constantly adds or removes from the ESCD data structure. Other parameters may need to be updated as a result. Regardless of the storage method chosen, the BIOS must know how to resolve any untimely interruption of a crucial system or BIOS function. The underlying mechanism for appeasing all these requirements is flash, and Intel's boot block flash is the solution for today's demands and tomorrow's inclinations.

Anyimi, 3.2

Figure 1 showed that either a 1-Mb or 2-Mb flash device can be used to implement BIOS. The choice of device depends on the contents of the BIOS and the level of sophistication desired in the design. The 1-Mb boot block flash memory is an established standard for implementing flash BIOS, not just for PnP. However, more BIOS space will be needed to support standardization of current features (like power management and virus aids) and impending future enhancements (like Windows 95 and Desktop Management Interfaces*, DMI). As consumers clamor for "more bang for the buck," more features and functions will be integrated into the standard PC. The migration beyond a 128-KB BIOS is inevitable. Already, some vendors have adopted code compression of BIOS in order to adhere to this 128-KB space limitation. This is a viable alternative that requires additional code decompression algorithms and consumes additional RAM space to store the full BIOS. Fortunately, Intel provides a secure means of code storage as well as a built-in growth path with its boot block architecture.

Anyimi, 5.1

1.1 maintaining a list of boot data used for booting a computer system;	Anyimi, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Anyimi discloses this claim limitation:

Today, PnP BIOS requirements may extend beyond 128 KB. A standard system BIOS consumes roughly 64 KB; PnP support is another 12 KB; automatic power management support adds 2 to 10 KB; PCI (Peripheral Components Interconnect) extensions hoard some 20 KB of the BIOS; and on-board VGA (Video Graphics Adapter) controllers utilize an extra 30 KB to 40 KB of space. The potential size of the BIOS for such a system is almost 150 KB. Although most of this code may be compressed, certain code segments cannot; besides even compression has its limits. Since new developments for the PC show no signs of abatement, it seems clear that a means of updating as well as expanding BIOS quickly and easily has become a necessity. Plug and Play is only one of the many technologies that can be implemented better with flash. As users realize the benefits of PnP, the demand generated will drive the number of Plug and Play systems upward.

Anyimi, 2.2

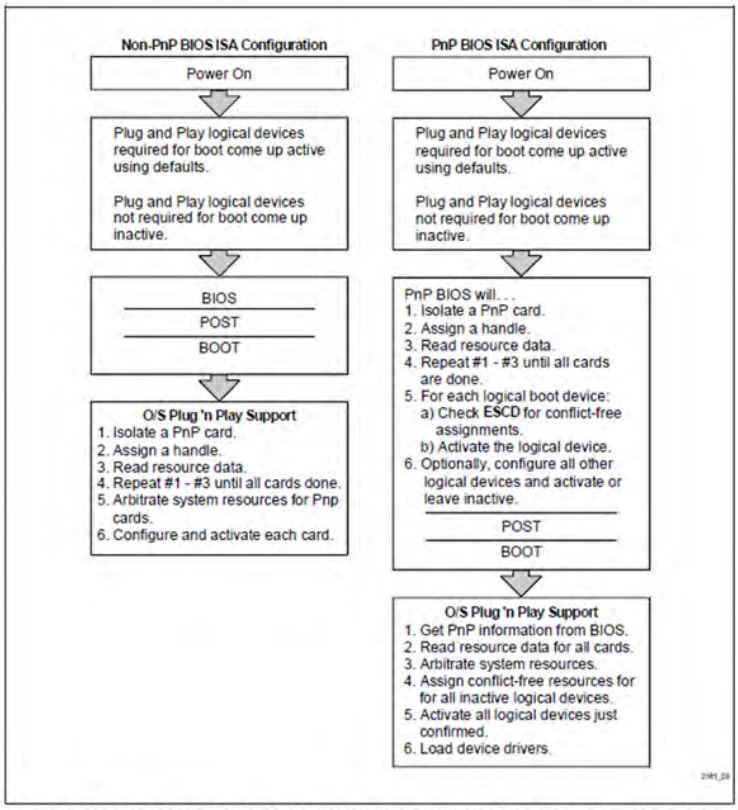


Figure 3. Possible PnP/Non-PnP BIOS ISA Add-In Card Configuration Flow. Note the importance of the ESCD in the Plug and Play Configuration Flow

Anyimi, Fig. 3

Furthermore, the PnP BIOS is capable of event management. Through its event management interfaces, the PnP BIOS can alert the system about new devices, like a notebook docking station, added or removed during runtime.

The POST procedure of the PnP BIOS identifies, tests, and configures the system before passing control to the operating system. The POST process must maintain previous POST compatibility, configure all legacy devices known to the PnP BIOS, arbitrate resources, initialize the IPL (Initial Program Load—this is how the operating system is launched), and support both PnP and non-PnP operating systems. Upon completion of the POST process, the BIOS attempts to have all necessary system devices initialized and enabled before the operating system is loaded; the PnP BIOS aspires further to provide the operating system a conflict-free environment in which to boot.

Anyimi, 3.2

Without an ESCD, the PnP BIOS must perform resource allocation as well as conflict detection and resolution each and every time the system is booted, and any locking of devices must be done by the supporting PnP operating system. Although the need for nonvolatile storage cannot be disputed, the amount of storage required depends on whether or not the PnP system needs real time updates. Ultimately, it will be decided by the methodology selected for resource management. A static or dynamic allocation scheme requires a fixed amount of nonvolatile memory for the ESCD and other BIOS parameters. A combined scheme for allocation constantly adds or removes from the ESCD data structure. Other parameters may need to be updated as a result. Regardless of the storage method chosen, the BIOS must know how to resolve any untimely interruption of a crucial system or BIOS function. The underlying mechanism for appeasing all these requirements is flash, and Intel's boot block flash is the solution for today's demands and tomorrow's inclinations.

Anyimi, 3.2

Figure 1 showed that either a 1-Mb or 2-Mb flash device can be used to implement BIOS. The choice of device depends on the contents of the BIOS and the level of sophistication desired in the design. The 1-Mb boot block flash memory is an established standard for implementing flash BIOS, not just for PnP. However, more BIOS space will be needed to support standardization of current features (like power management and virus aids) and impending future enhancements (like Windows 95 and Desktop Management Interfaces*, DMI). As consumers clamor for "more bang for the buck," more features and functions will be integrated into the standard PC. The migration beyond a 128-KB BIOS is inevitable. Already, some vendors have adopted code compression of BIOS in order to adhere to this 128-KB space limitation. This is a viable alternative that requires additional code decompression algorithms and consumes additional RAM space to store the full BIOS. Fortunately, Intel provides a secure means of code storage as well as a built-in growth path with its boot block architecture.

Anyimi, 5.1

1.2 initializing a central processing unit of the computer system;	Anyimi, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Anyimi discloses this claim limitation:

3.1 PnP Components

For a system to be fully PnP-compliant, four basic elements are required:

1. System/PnP BIOS
2. PnP operating system
3. PnP hardware devices
4. PnP application software

Anyimi, 3.1

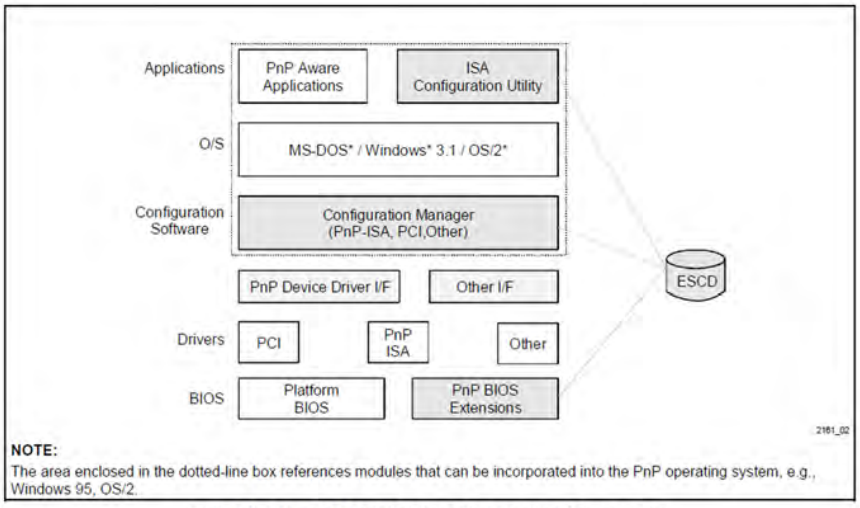


Figure 2. Plug and Play Software Architecture Components.
Note the many layers that depend on the ESCD. The parameter blocks of the boot block flash memory enable this nonvolatile storage capability of Plug and Play.

Anyimi, Fig. 2

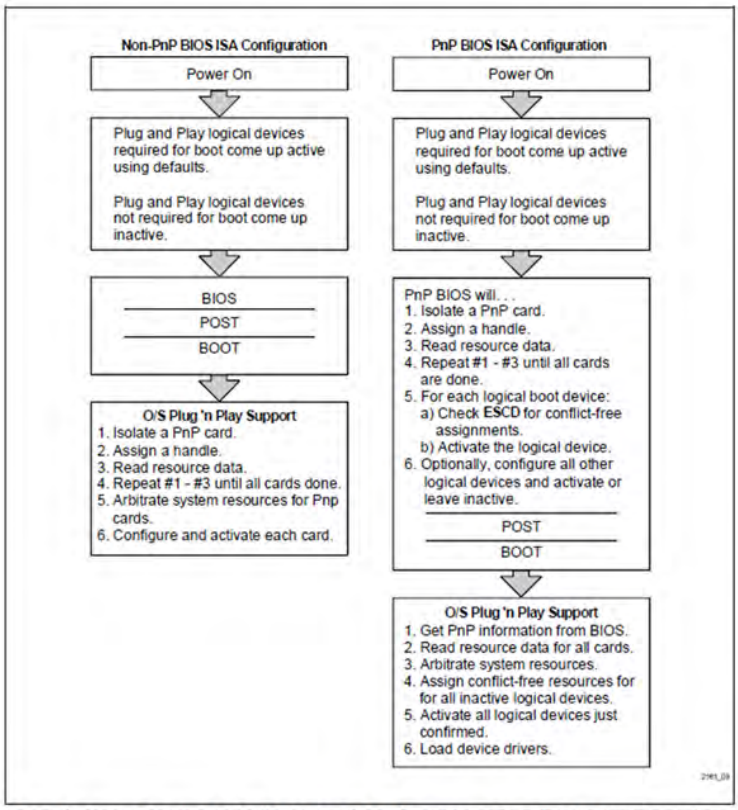


Figure 3. Possible PnP/Non-PnP BIOS ISA Add-In Card Configuration Flow. Note the importance of the ESCD in the Plug and Play Configuration Flow

Anyimi, Fig. 3

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Anyimi, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Anyimi discloses this claim limitation:

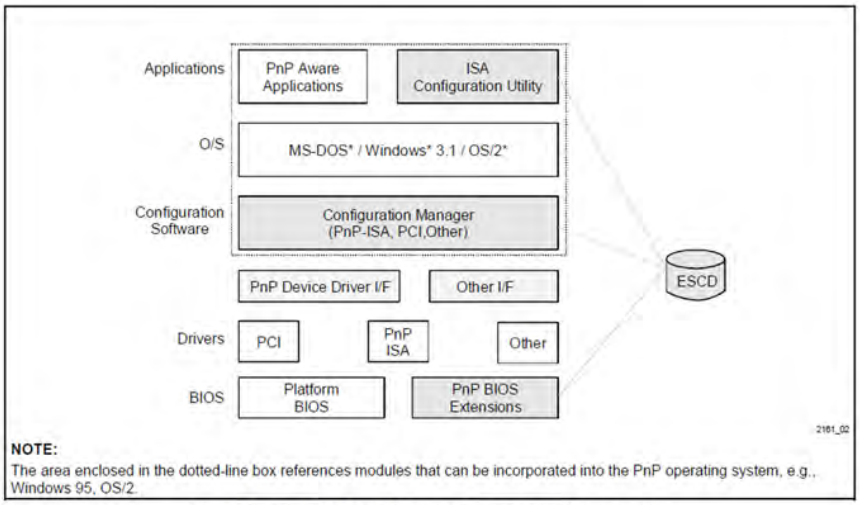


Figure 2. Plug and Play Software Architecture Components.
Note the many layers that depend on the ESCD. The parameter blocks of the boot block flash memory enable this nonvolatile storage capability of Plug and Play.

Anyimi, Fig. 2

Today, PnP BIOS requirements may extend beyond 128 KB. A standard system BIOS consumes roughly 64 KB; PnP support is another 12 KB; automatic power management support adds 2 to 10 KB; PCI (Peripheral Components Interconnect) extensions hoard some 20 KB of the BIOS; and on-board VGA (Video Graphics Adapter) controllers utilize an extra 30 KB to 40 KB of space. The potential size of the BIOS for such a system is almost 150 KB. Although most of this code may be compressed, certain code segments cannot; besides even compression has its limits. Since new developments for the PC show no signs of abatement, it seems clear that a means of updating as well as expanding BIOS quickly and easily has become a necessity. Plug and Play is only one of the many technologies that can be implemented better with flash. As users realize the benefits of PnP, the demand generated will drive the number of Plug and Play systems upward.

Anyimi, 2.2

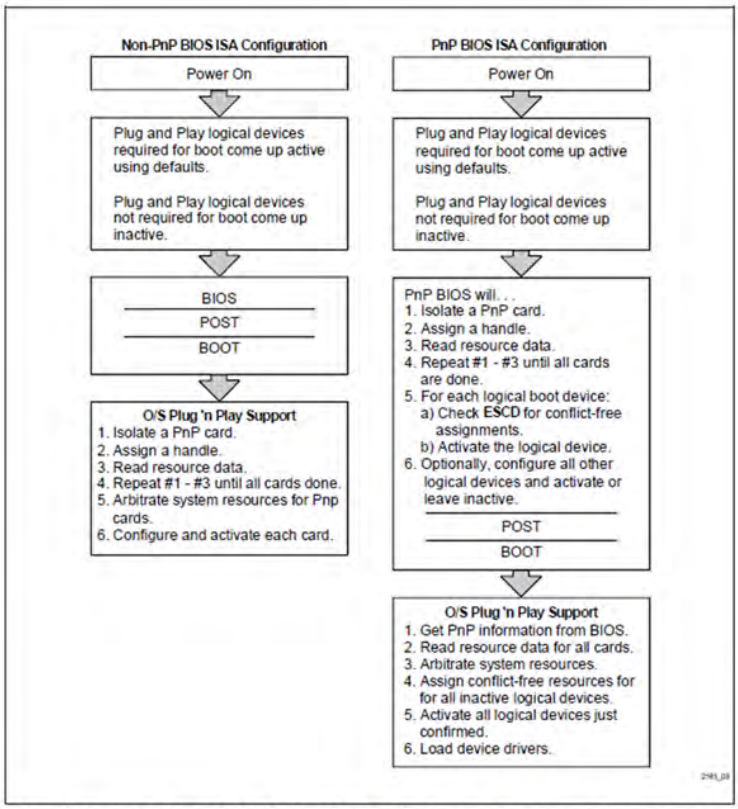


Figure 3. Possible PnP/Non-PnP BIOS ISA Add-In Card Configuration Flow. Note the importance of the ESCD in the Plug and Play Configuration Flow

Anyimi, Fig. 3

Furthermore, the PnP BIOS is capable of event management. Through its event management interfaces, the PnP BIOS can alert the system about new devices, like a notebook docking station, added or removed during runtime.

The POST procedure of the PnP BIOS identifies, tests, and configures the system before passing control to the operating system. The POST process must maintain previous POST compatibility, configure all legacy devices known to the PnP BIOS, arbitrate resources, initialize the IPL (Initial Program Load—this is how the operating system is launched), and support both PnP and non-PnP operating systems. Upon completion of the POST process, the BIOS attempts to have all necessary system devices initialized and enabled before the operating system is loaded; the PnP BIOS aspires further to provide the operating system a conflict-free environment in which to boot.

Anyimi, 3.2

Without an ESCD, the PnP BIOS must perform resource allocation as well as conflict detection and resolution each and every time the system is booted, and any locking of devices must be done by the supporting PnP operating system. Although the need for nonvolatile storage cannot be disputed, the amount of storage required depends on whether or not the PnP system needs real time updates. Ultimately, it will be decided by the methodology selected for resource management. A static or dynamic allocation scheme requires a fixed amount of nonvolatile memory for the ESCD and other BIOS parameters. A combined scheme for allocation constantly adds or removes from the ESCD data structure. Other parameters may need to be updated as a result. Regardless of the storage method chosen, the BIOS must know how to resolve any untimely interruption of a crucial system or BIOS function. The underlying mechanism for appeasing all these requirements is flash, and Intel's boot block flash is the solution for today's demands and tomorrow's inclinations.

Anyimi, 3.2

Figure 1 showed that either a 1-Mb or 2-Mb flash device can be used to implement BIOS. The choice of device depends on the contents of the BIOS and the level of sophistication desired in the design. The 1-Mb boot block flash memory is an established standard for implementing flash BIOS, not just for PnP. However, more BIOS space will be needed to support standardization of current features (like power management and virus aids) and impending future enhancements (like Windows 95 and Desktop Management Interfaces*, DMI). As consumers clamor for "more bang for the buck," more features and functions will be integrated into the standard PC. The migration beyond a 128-KB BIOS is inevitable. Already, some vendors have adopted code compression of BIOS in order to adhere to this 128-KB space limitation. This is a viable alternative that requires additional code decompression algorithms and consumes additional RAM space to store the full BIOS. Fortunately, Intel provides a secure means of code storage as well as a built-in growth path with its boot block architecture.

Anyimi, 5.1

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Anyimi, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Anyimi discloses this claim limitation:

Today, PnP BIOS requirements may extend beyond 128 KB. A standard system BIOS consumes roughly 64 KB; PnP support is another 12 KB; automatic power management support adds 2 to 10 KB; PCI (Peripheral Components Interconnect) extensions hoard some 20 KB of the BIOS; and on-board VGA (Video Graphics Adapter) controllers utilize an extra 30 KB to 40 KB of space. The potential size of the BIOS for such a system is almost 150 KB. Although most of this code may be compressed, certain code segments cannot; besides even compression has its limits. Since new developments for the PC show no signs of abatement, it seems clear that a means of updating as well as expanding BIOS quickly and easily has become a necessity. Plug and Play is only one of the many technologies that can be implemented better with flash. As users realize the benefits of PnP, the demand generated will drive the number of Plug and Play systems upward.

Anyimi, 2.2

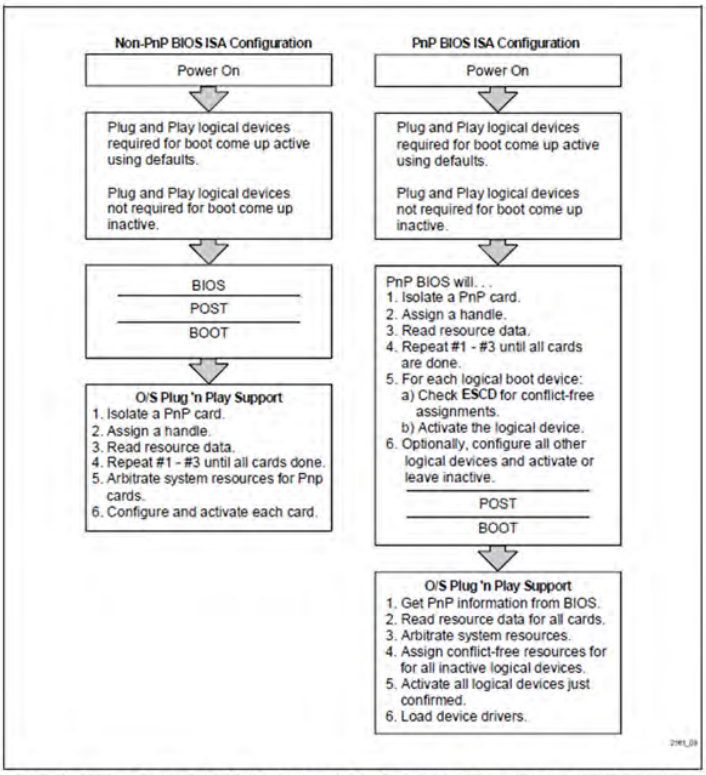


Figure 3. Possible PnP/Non-PnP BIOS ISA Add-In Card Configuration Flow. Note the importance of the ESCD in the Plug and Play Configuration Flow

Anyimi, Fig. 3

Furthermore, the PnP BIOS is capable of event management. Through its event management interfaces, the PnP BIOS can alert the system about new devices, like a notebook docking station, added or removed during runtime.

The POST procedure of the PnP BIOS identifies, tests, and configures the system before passing control to the operating system. The POST process must maintain previous POST compatibility, configure all legacy devices known to the PnP BIOS, arbitrate resources, initialize the IPL (Initial Program Load—this is how the operating system is launched), and support both PnP and non-PnP operating systems. Upon completion of the POST process, the BIOS attempts to have all necessary system devices initialized and enabled before the operating system is loaded; the PnP BIOS aspires further to provide the operating system a conflict-free environment in which to boot.

Anyimi, 3.2

Without an ESCD, the PnP BIOS must perform resource allocation as well as conflict detection and resolution each and every time the system is booted, and any locking of devices must be done by the supporting PnP operating system. Although the need for nonvolatile storage cannot be disputed, the amount of storage required depends on whether or not the PnP system needs real time updates. Ultimately, it will be decided by the methodology selected for resource management. A static or dynamic allocation scheme requires a fixed amount of nonvolatile memory for the ESCD and other BIOS parameters. A combined scheme for allocation constantly adds or removes from the ESCD data structure. Other parameters may need to be updated as a result. Regardless of the storage method chosen, the BIOS must know how to resolve any untimely interruption of a crucial system or BIOS function. The underlying mechanism for appeasing all these requirements is flash, and Intel's boot block flash is the solution for today's demands and tomorrow's inclinations.

Anyimi, 3.2

Anyimi Claim 1.4

"servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

Figure 1 showed that either a 1-Mb or 2-Mb flash device can be used to implement BIOS. The choice of device depends on the contents of the BIOS and the level of sophistication desired in the design. The 1-Mb boot block flash memory is an established standard for implementing flash BIOS, not just for PnP. However, more BIOS space will be needed to support standardization of current features (like power management and virus aids) and impending future enhancements (like Windows 95 and Desktop Management Interfaces*, DMI). As consumers clamor for "more bang for the buck," more features and functions will be integrated into the standard PC. The migration beyond a 128-KB BIOS is inevitable. Already, some vendors have adopted code compression of BIOS in order to adhere to this 128-KB space limitation. This is a viable alternative that requires additional code decompression algorithms and consumes additional RAM space to store the full BIOS. Fortunately, Intel provides a secure means of code storage as well as a built-in growth path with its boot block architecture.

Anyimi, 5.1

Anyimi Claim 1.4

"servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Anyimi, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Anyimi discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

Perhaps you are wondering why flash is the medium of choice advocated in this application note as the means for storing system BIOS, particularly for Plug and Play? From a PC user's perspective, a PnP flash BIOS enables users to install new hardware without having to call the support number. This translates to ease-of-use:

- Easily updatable code assuring optimal BIOS performance
- No need to edit the CONFIG.SYS file.
- No need to determine system type and match, somehow, to jumper settings.
- No need to investigate available system resources and muddle through reassigning them.
- No need to fiddle with system memory reallocation.
- No need to buy a new system every time something changes (increases system's useful lifespan).

Anyimi, 2.0

For a system to be fully PnP-compliant, four basic elements are required:

1. System/PnP BIOS
2. PnP operating system
3. PnP hardware devices
4. PnP application software

Anyimi, 3.1

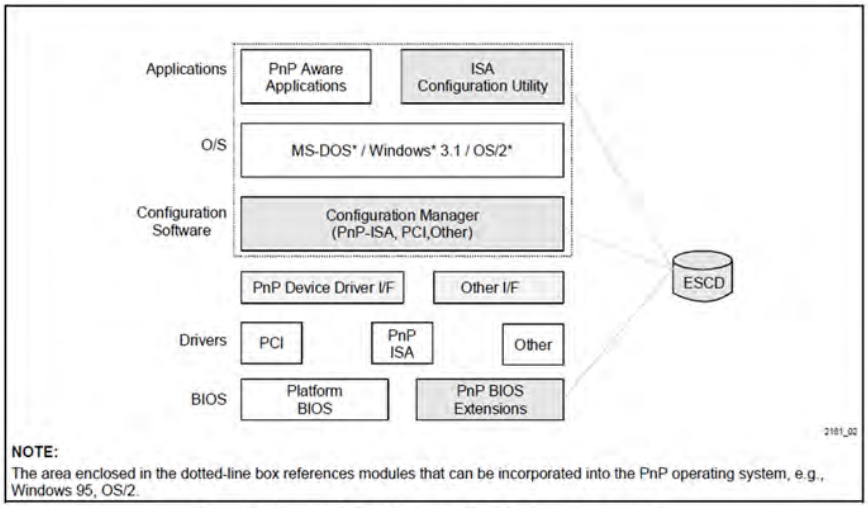


Figure 2. Plug and Play Software Architecture Components. Note the many layers that depend on the ESCD. The parameter blocks of the boot block flash memory enable this nonvolatile storage capability of Plug and Play.

Anyimi, Fig. 2

The Intel boot block (BB) flash memory products are particularly well suited for BIOS applications. Boot block flash is segmented into a lockable boot block, two parameter blocks and one or more main blocks. All boot

block devices are manufactured on Intel's ETOX™ flash memory process technology. An on-chip Write State Machine (WSM) provides automated program and erase algorithms with an SRAM-compatible write interface. The key feature of the boot block architecture that differentiates it from other flash memories is its hardware-lockable boot block, which allows system recovery from fatal crashes. Additional features of boot block flash include:

- Hardware write protection via pin
- Hardware locking of boot block
- Hardware pin for system reset during write
- High performance reads (for speedy access to data)
- Deep power-down mode (a key feature for "green" PCs)
- Extended cycling capability (100,000 block/erase cycles)

Anyimi, 4.0

Anyimi Claim 2

"The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Anyimi, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Anyimi discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

The following items are required to have an ISW capable system for updating the PnP BIOS:

- Microprocessor or controller (to control the reprogramming process)
- PnP BIOS boot code, communications software, and PnP BIOS update algorithm
- Data import capability (floppy disk, serial, network, etc.)
- Vpp generator or regulator (12V products only)

Anyimi, 5.3.2.1

4. The method of claim 1, further comprising updating the list of boot data.	Anyimi, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Anyimi discloses this limitation:

See Claim 1.1 above.

See also

Anyimi discloses this claim limitation:

Plug and Play can make the usual experience of adding new functionality to an existing system as easy as:

1. Turn the system off.
2. Insert the new device.
3. Turn the system on.

PnP flash BIOS can also extend the life of the PC. By enabling simple updates to the BIOS (simply insert the upgrade disk and the software programs the new BIOS), the user can get more out of their PC investment.

Anyimi, 1.0

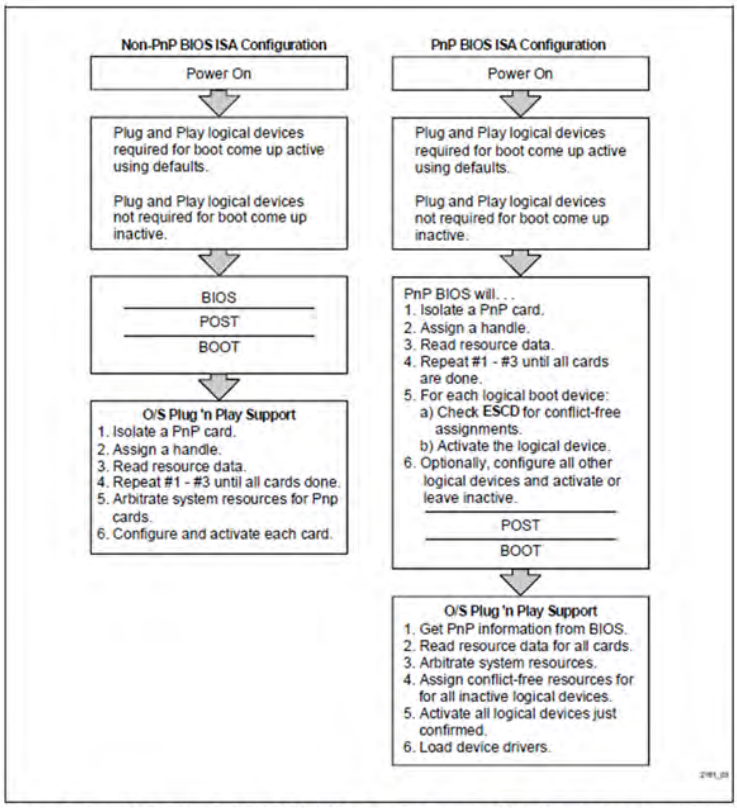


Figure 3. Possible PnP/Non-PnP BIOS ISA Add-In Card Configuration Flow. Note the importance of the ESCD in the Plug and Play Configuration Flow

Anyimi, Fig. 3

Furthermore, the PnP BIOS is capable of event management. Through its event management interfaces, the PnP BIOS can alert the system about new devices, like a notebook docking station, added or removed during runtime.

The POST procedure of the PnP BIOS identifies, tests, and configures the system before passing control to the operating system. The POST process must maintain previous POST compatibility, configure all legacy devices known to the PnP BIOS, arbitrate resources, initialize the IPL (Initial Program Load—this is how the operating system is launched), and support both PnP and non-PnP operating systems. Upon completion of the POST process, the BIOS attempts to have all necessary system devices initialized and enabled before the operating system is loaded; the PnP BIOS aspires further to provide the operating system a conflict-free environment in which to boot.

Anyimi, 3.2

Without an ESCD, the PnP BIOS must perform resource allocation as well as conflict detection and resolution each and every time the system is booted, and any locking of devices must be done by the supporting PnP operating system. Although the need for nonvolatile storage cannot be disputed, the amount of storage required depends on whether or not the PnP system needs real time updates. Ultimately, it will be decided by the methodology selected for resource management. A static or dynamic allocation scheme requires a fixed amount of nonvolatile memory for the ESCD and other BIOS parameters. A combined scheme for allocation constantly adds or removes from the ESCD data structure. Other parameters may need to be updated as a result. Regardless of the storage method chosen, the BIOS must know how to resolve any untimely interruption of a crucial system or BIOS function. The underlying mechanism for appeasing all these requirements is flash, and Intel's boot block flash is the solution for today's demands and tomorrow's inclinations.

Anyimi, 3.2

Figure 1 showed that either a 1-Mb or 2-Mb flash device can be used to implement BIOS. The choice of device depends on the contents of the BIOS and the level of sophistication desired in the design. The 1-Mb boot block flash memory is an established standard for implementing flash BIOS, not just for PnP. However, more BIOS space will be needed to support standardization of current features (like power management and virus aids) and impending future enhancements (like Windows 95 and Desktop Management Interfaces*, DMI). As consumers clamor for "more bang for the buck," more features and functions will be integrated into the standard PC. The migration beyond a 128-KB BIOS is inevitable. Already, some vendors have adopted code compression of BIOS in order to adhere to this 128-KB space limitation. This is a viable alternative that requires additional code decompression algorithms and consumes additional RAM space to store the full BIOS. Fortunately, Intel provides a secure means of code storage as well as a built-in growth path with its boot block architecture.

Anyimi, 5.1

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Anyimi, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Anyimi, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Anyimi, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Anyimi, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Anyimi discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

The following items are required to have an ISW capable system for updating the PnP BIOS:

- Microprocessor or controller (to control the reprogramming process)
- PnP BIOS boot code, communications software, and PnP BIOS update algorithm
- Data import capability (floppy disk, serial, network, etc.)
- Vpp generator or regulator (12V products only)

Anyimi, 5.3.2.1

7.2 a cache memory device; and;	Anyimi, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Anyimi, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Anyimi, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Anyimi, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Anyimi, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Anyimi discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

Perhaps you are wondering why flash is the medium of choice advocated in this application note as the means for storing system BIOS, particularly for Plug and Play? From a PC user's perspective, a PnP flash BIOS enables users to install new hardware without having to call the support number. This translates to ease-of-use:

- Easily updatable code assuring optimal BIOS performance
- No need to edit the CONFIG.SYS file.
- No need to determine system type and match, somehow, to jumper settings.
- No need to investigate available system resources and muddle through reassigning them.
- No need to fiddle with system memory reallocation.
- No need to buy a new system every time something changes (increases system's useful lifespan).

Anyimi, 2.0

For a system to be fully PnP-compliant, four basic elements are required:

1. System/PnP BIOS
2. PnP operating system
3. PnP hardware devices
4. PnP application software

Anyimi, 3.1

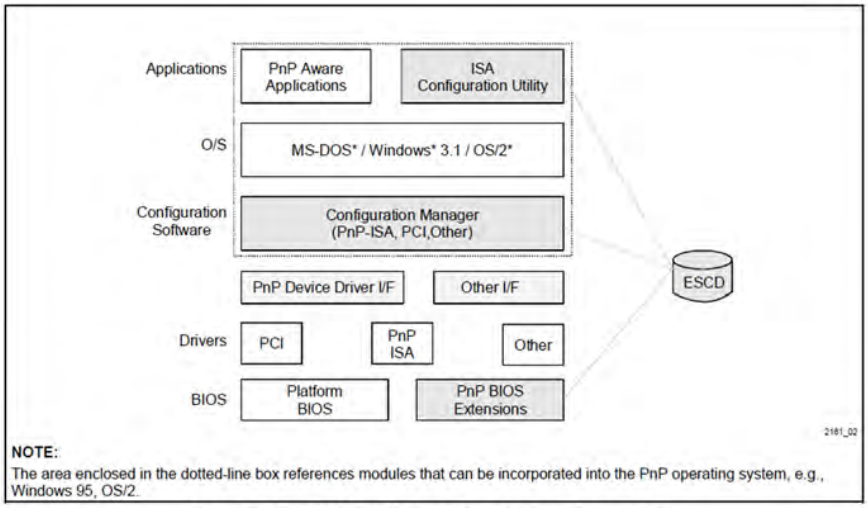


Figure 2. Plug and Play Software Architecture Components. Note the many layers that depend on the ESCD. The parameter blocks of the boot block flash memory enable this nonvolatile storage capability of Plug and Play.

Anyimi, Fig. 2

The Intel boot block (BB) flash memory products are particularly well suited for BIOS applications. Boot block flash is segmented into a lockable boot block, two parameter blocks and one or more main blocks. All boot

block devices are manufactured on Intel's ETOX™ flash memory process technology. An on-chip Write State Machine (WSM) provides automated program and erase algorithms with an SRAM-compatible write interface. The key feature of the boot block architecture that differentiates it from other flash memories is its hardware-lockable boot block, which allows system recovery from fatal crashes. Additional features of boot block flash include:

- Hardware write protection via pin
- Hardware locking of boot block
- Hardware pin for system reset during write
- High performance reads (for speedy access to data)
- Deep power-down mode (a key feature for "green" PCs)
- Extended cycling capability (100,000 block/erase cycles)

Anyimi, 4.0

9.1 maintaining a list of application data associated with an application program;	Anyimi, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Anyimi, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Anyimi, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Anyimi discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

Perhaps you are wondering why flash is the medium of choice advocated in this application note as the means for storing system BIOS, particularly for Plug and Play? From a PC user's perspective, a PnP flash BIOS enables users to install new hardware without having to call the support number. This translates to ease-of-use:

- Easily updatable code assuring optimal BIOS performance
- No need to edit the CONFIG.SYS file.
- No need to determine system type and match, somehow, to jumper settings.
- No need to investigate available system resources and muddle through reassigning them.
- No need to fiddle with system memory reallocation.
- No need to buy a new system every time something changes (increases system's useful lifespan).

Anyimi, 2.0

For a system to be fully PnP-compliant, four basic elements are required:

1. System/PnP BIOS
2. PnP operating system
3. PnP hardware devices
4. PnP application software

Anyimi, 3.1

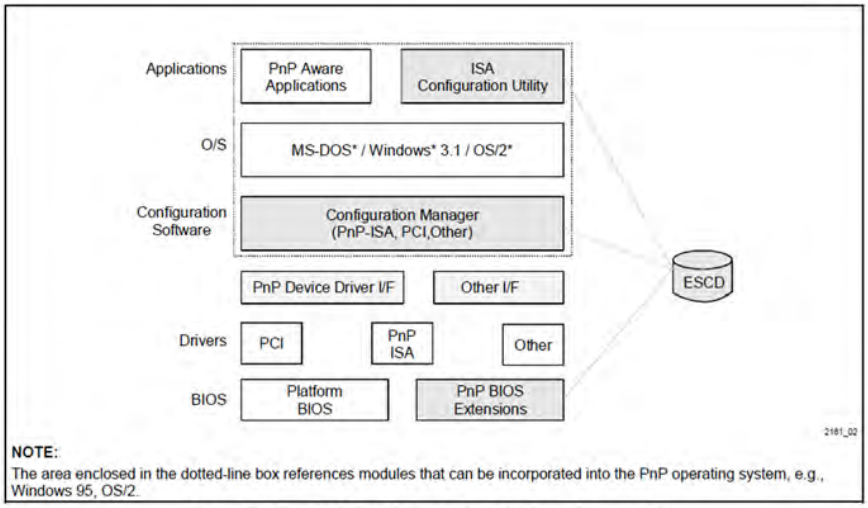


Figure 2. Plug and Play Software Architecture Components. Note the many layers that depend on the ESCD. The parameter blocks of the boot block flash memory enable this nonvolatile storage capability of Plug and Play.

Anyimi, Fig. 2

The Intel boot block (BB) flash memory products are particularly well suited for BIOS applications. Boot block flash is segmented into a lockable boot block, two parameter blocks and one or more main blocks. All boot

block devices are manufactured on Intel's ETOX™ flash memory process technology. An on-chip Write State Machine (WSM) provides automated program and erase algorithms with an SRAM-compatible write interface. The key feature of the boot block architecture that differentiates it from other flash memories is its hardware-lockable boot block, which allows system recovery from fatal crashes. Additional features of boot block flash include:

- Hardware write protection via pin
- Hardware locking of boot block
- Hardware pin for system reset during write
- High performance reads (for speedy access to data)
- Deep power-down mode (a key feature for "green" PCs)
- Extended cycling capability (100,000 block/erase cycles)

Anyimi, 4.0

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Anyimi, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Anyimi discloses this limitation:

Plug and Play can make the usual experience of adding new functionality to an existing system as easy as:

1. Turn the system off.
2. Insert the new device.
3. Turn the system on.

PnP flash BIOS can also extend the life of the PC. By enabling simple updates to the BIOS (simply insert the upgrade disk and the software programs the new BIOS), the user can get more out of their PC investment.

Anyimi, 1.0

3.1 PnP Components

For a system to be fully PnP-compliant, four basic elements are required:

1. System/PnP BIOS
2. PnP operating system
3. PnP hardware devices
4. PnP application software

Anyimi, 3.1

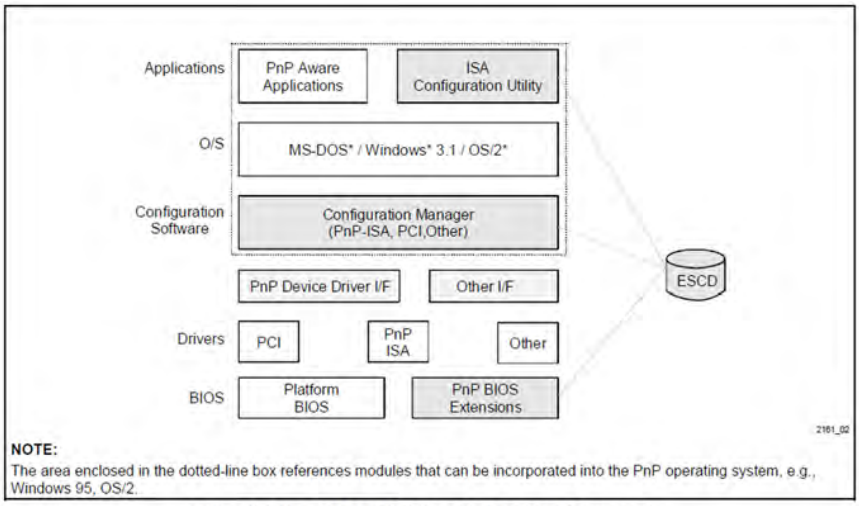


Figure 2. Plug and Play Software Architecture Components.
Note the many layers that depend on the ESCD. The parameter blocks of the boot block flash memory enable this nonvolatile storage capability of Plug and Play.

Anyimi, Fig. 2

Today, PnP BIOS requirements may extend beyond 128 KB. A standard system BIOS consumes roughly 64 KB; PnP support is another 12 KB; automatic power management support adds 2 to 10 KB; PCI (Peripheral Components Interconnect) extensions hoard some 20 KB of the BIOS; and on-board VGA (Video Graphics Adapter) controllers utilize an extra 30 KB to 40 KB of space. The potential size of the BIOS for such a system is almost 150 KB. Although most of this code may be compressed, certain code segments cannot; besides even compression has its limits. Since new developments for the PC show no signs of abatement, it seems clear that a means of updating as well as expanding BIOS quickly and easily has become a necessity. Plug and Play is only one of the many technologies that can be implemented better with flash. As users realize the benefits of PnP, the demand generated will drive the number of Plug and Play systems upward.

Anyimi, 2.2

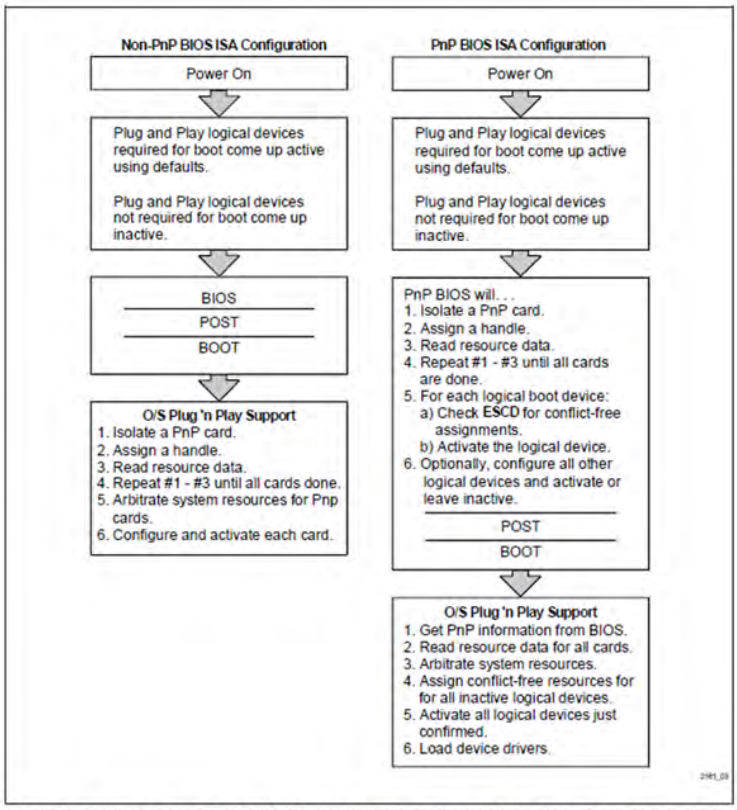


Figure 3. Possible PnP/Non-PnP BIOS ISA Add-In Card Configuration Flow. Note the importance of the ESCD in the Plug and Play Configuration Flow

Anyimi, Fig. 3

Furthermore, the PnP BIOS is capable of event management. Through its event management interfaces, the PnP BIOS can alert the system about new devices, like a notebook docking station, added or removed during runtime.

The POST procedure of the PnP BIOS identifies, tests, and configures the system before passing control to the operating system. The POST process must maintain previous POST compatibility, configure all legacy devices known to the PnP BIOS, arbitrate resources, initialize the IPL (Initial Program Load—this is how the operating system is launched), and support both PnP and non-PnP operating systems. Upon completion of the POST process, the BIOS attempts to have all necessary system devices initialized and enabled before the operating system is loaded; the PnP BIOS aspires further to provide the operating system a conflict-free environment in which to boot.

Anyimi, 3.2

Anyimi Claim 10

"The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device"

Without an ESCD, the PnP BIOS must perform resource allocation as well as conflict detection and resolution each and every time the system is booted, and any locking of devices must be done by the supporting PnP operating system. Although the need for nonvolatile storage cannot be disputed, the amount of storage required depends on whether or not the PnP system needs real time updates. Ultimately, it will be decided by the methodology selected for resource management. A static or dynamic allocation scheme requires a fixed amount of nonvolatile memory for the ESCD and other BIOS parameters. A combined scheme for allocation constantly adds or removes from the ESCD data structure. Other parameters may need to be updated as a result. Regardless of the storage method chosen, the BIOS must know how to resolve any untimely interruption of a crucial system or BIOS function. The underlying mechanism for appeasing all these requirements is flash, and Intel's boot block flash is the solution for today's demands and tomorrow's inclinations.

Anyimi, 3.2

Figure 1 showed that either a 1-Mb or 2-Mb flash device can be used to implement BIOS. The choice of device depends on the contents of the BIOS and the level of sophistication desired in the design. The 1-Mb boot block flash memory is an established standard for implementing flash BIOS, not just for PnP. However, more BIOS space will be needed to support standardization of current features (like power management and virus aids) and impending future enhancements (like Windows 95 and Desktop Management Interfaces*, DMI). As consumers clamor for "more bang for the buck," more features and functions will be integrated into the standard PC. The migration beyond a 128-KB BIOS is inevitable. Already, some vendors have adopted code compression of BIOS in order to adhere to this 128-KB space limitation. This is a viable alternative that requires additional code decompression algorithms and consumes additional RAM space to store the full BIOS. Fortunately, Intel provides a secure means of code storage as well as a built-in growth path with its boot block architecture.

Anyimi, 5.1

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Anyimi, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Anyimi discloses this limitation:

Today, PnP BIOS requirements may extend beyond 128 KB. A standard system BIOS consumes roughly 64 KB; PnP support is another 12 KB; automatic power management support adds 2 to 10 KB; PCI (Peripheral Components Interconnect) extensions hoard some 20 KB of the BIOS; and on-board VGA (Video Graphics Adapter) controllers utilize an extra 30 KB to 40 KB of space. The potential size of the BIOS for such a system is almost 150 KB. Although most of this code may be compressed, certain code segments cannot; besides even compression has its limits. Since new developments for the PC show no signs of abatement, it seems clear that a means of updating as well as expanding BIOS quickly and easily has become a necessity. Plug and Play is only one of the many technologies that can be implemented better with flash. As users realize the benefits of PnP, the demand generated will drive the number of Plug and Play systems upward.

Anyimi, 2.2

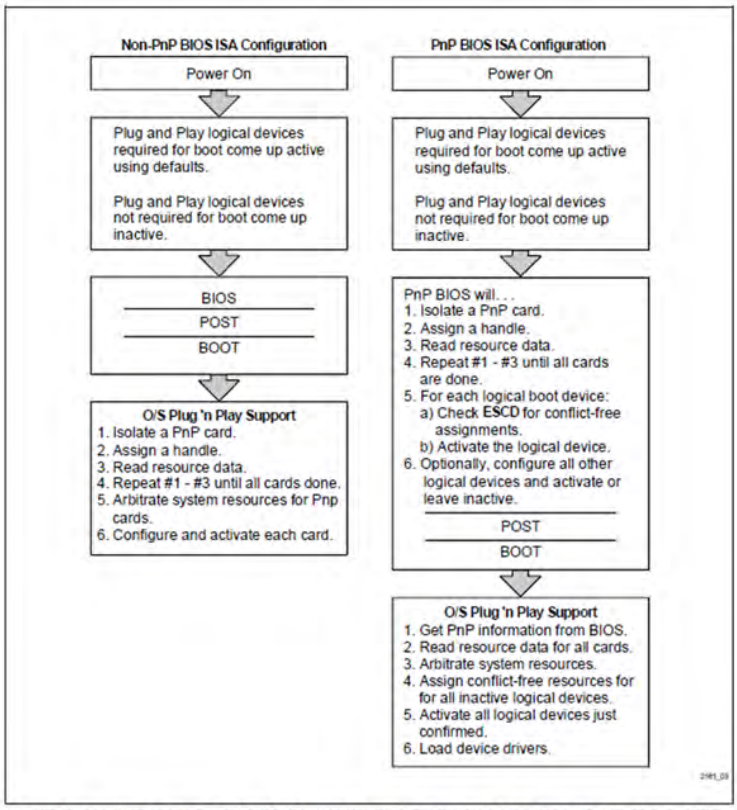


Figure 3. Possible PnP/Non-PnP BIOS ISA Add-In Card Configuration Flow. Note the importance of the ESCD in the Plug and Play Configuration Flow

Anyimi, Fig. 3

Furthermore, the PnP BIOS is capable of event management. Through its event management interfaces, the PnP BIOS can alert the system about new devices, like a notebook docking station, added or removed during runtime.

The POST procedure of the PnP BIOS identifies, tests, and configures the system before passing control to the operating system. The POST process must maintain previous POST compatibility, configure all legacy devices known to the PnP BIOS, arbitrate resources, initialize the IPL (Initial Program Load—this is how the operating system is launched), and support both PnP and non-PnP operating systems. Upon completion of the POST process, the BIOS attempts to have all necessary system devices initialized and enabled before the operating system is loaded; the PnP BIOS aspires further to provide the operating system a conflict-free environment in which to boot.

Anyimi, 3.2

Without an ESCD, the PnP BIOS must perform resource allocation as well as conflict detection and resolution each and every time the system is booted, and any locking of devices must be done by the supporting PnP operating system. Although the need for nonvolatile storage cannot be disputed, the amount of storage required depends on whether or not the PnP system needs real time updates. Ultimately, it will be decided by the methodology selected for resource management. A static or dynamic allocation scheme requires a fixed amount of nonvolatile memory for the ESCD and other BIOS parameters. A combined scheme for allocation constantly adds or removes from the ESCD data structure. Other parameters may need to be updated as a result. Regardless of the storage method chosen, the BIOS must know how to resolve any untimely interruption of a crucial system or BIOS function. The underlying mechanism for appeasing all these requirements is flash, and Intel's boot block flash is the solution for today's demands and tomorrow's inclinations.

Anyimi, 3.2

Figure 1 showed that either a 1-Mb or 2-Mb flash device can be used to implement BIOS. The choice of device depends on the contents of the BIOS and the level of sophistication desired in the design. The 1-Mb boot block flash memory is an established standard for implementing flash BIOS, not just for PnP. However, more BIOS space will be needed to support standardization of current features (like power management and virus aids) and impending future enhancements (like Windows 95 and Desktop Management Interfaces*, DMI). As consumers clamor for "more bang for the buck," more features and functions will be integrated into the standard PC. The migration beyond a 128-KB BIOS is inevitable. Already, some vendors have adopted code compression of BIOS in order to adhere to this 128-KB space limitation. This is a viable alternative that requires additional code decompression algorithms and consumes additional RAM space to store the full BIOS. Fortunately, Intel provides a secure means of code storage as well as a built-in growth path with its boot block architecture.

Anyimi, 5.1

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Anyimi, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Anyimi, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Anyimi, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Anyimi, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 1.3, 1.4, and 15 above.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Anyimi, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Anyimi, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Anyimi, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Anyimi, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Anyimi, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Anyimi, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Anyimi, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Anyimi, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Anyimi, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Anyimi, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Anyimi, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Anyimi, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Anyimi, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Anyimi, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Anyimi, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Anyimi, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Anyimi, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Anyimi, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 15 and 27 above	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Anyimi, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Anyimi discloses this limitation: *See* Claims 16 and 27 above	

Appendix A26
Invalidity of U.S. Patent 7,181,608 based on Bennett

The publication D. Bennett, "Booting Linux from EPROM," Linux Journal, January 1997 ("Bennett") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A26
Invalidity of U.S. Patent 7,181,608 based on Bennett

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Bennett, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Bennett discloses this claim limitation:

System Operation

For booting, two options were considered:

- booting DOS, then running the loadlin program (to load Linux) from autoexec.bat
- installing LILO and booting Linux directly

The advantage of the second option would be a slightly shorter boot time. However, we implemented the first option, because we wanted to use a programmable keyboard—the software for programming the keyboard runs under DOS.

Bennett, 1

When deciding how to implement the EPROM device driver, the first idea was to create an image of a disk in the EPROM. This would provide a RAM disk of the same size as the EPROM, 3.5MB in this case (the DOS portion of the SSD takes 1/2 MB). Instead, to allow a larger RAM disk, a compressed disk image is used. The compression used is simple—any sectors which are identical are only stored once. The primary advantage this gives is blank areas of the disk image don't need to take up EPROM space. Listing 1 shows the SSD disk compression used.

Bennett, 1

The boot sequence is:

- Power up and run memory tests
- load DOS which executes AUTOEXEC.BAT
  - run the keyboard programming application
  - run loadlin—this reads a linux kernel from the DOS disk & executes it
- the linux kernel takes over
- load the RAM disk from the EPROM disk
- switch the root file system to the RAM disk
- init will read inittab which executes dboot instead of getty
- the operator interface application is started

Bennett, 2

The first phase of disk image development was identifying the required and the desired items. The first step was to come up with a minimal system and then add the items required for the operator interface. Not being a Unix expert, coming up with the minimal system ended up being something of a trial and error process. I started with what I thought was needed, then tried running it. When an error occurred because of a missing program or library, that file was added. This process went on until the system ran happily.

The bulk of this was done by copying files from the "full" Linux partition to the 6MB partition, booting DOS and using the loadlin line:

```
loadlin zimage root=/dev/hda2 ro
```

Bennett, 3

Once the system was fairly stable, the 6MB partition was loaded into the RAM disk. This is very similar to how the RAM disk is loaded from EPROM, but development went faster since EPROMs weren't being programmed. To test the system without programming EPROMs, the system booted DOS and ran loadlin with the line:

```
loadlin zimage root=/dev/hda2 ramdisk=6144 ro
```

Because of the modification to ramdisk.c, the /dev/hda2 disk image is loaded into the RAM disk, then the root file system is switched to the RAM disk. The process of refining the disk image continues until everything is "perfect".

Bennett, 4

1.1 maintaining a list of boot data used for booting a computer system;	Bennett, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Bennett discloses this claim limitation:

System Operation

For booting, two options were considered:

- booting DOS, then running the loadlin program (to load Linux) from autoexec.bat
- installing LILO and booting Linux directly

The advantage of the second option would be a slightly shorter boot time. However, we implemented the first option, because we wanted to use a programmable keyboard—the software for programming the keyboard runs under DOS.

Bennett, 1

When deciding how to implement the EPROM device driver, the first idea was to create an image of a disk in the EPROM. This would provide a RAM disk of the same size as the EPROM, 3.5MB in this case (the DOS portion of the SSD takes 1/2 MB). Instead, to allow a larger RAM disk, a compressed disk image is used. The compression used is simple—any sectors which are identical are only stored once. The primary advantage this gives is blank areas of the disk image don't need to take up EPROM space. Listing 1 shows the SSD disk compression used.

Bennett, 1

The boot sequence is:

- Power up and run memory tests
- load DOS which executes AUTOEXEC.BAT
  - run the keyboard programming application
  - run loadlin—this reads a linux kernel from the DOS disk & executes it
- the linux kernel takes over
- load the RAM disk from the EPROM disk
- switch the root file system to the RAM disk
- init will read inittab which executes dboot instead of getty
- the operator interface application is started

Bennett, 2

The first phase of disk image development was identifying the required and the desired items. The first step was to come up with a minimal system and then add the items required for the operator interface. Not being a Unix expert, coming up with the minimal system ended up being something of a trial and error process. I started with what I thought was needed, then tried running it. When an error occurred because of a missing program or library, that file was added. This process went on until the system ran happily.

The bulk of this was done by copying files from the "full" Linux partition to the 6MB partition, booting DOS and using the loadlin line:

```
loadlin zimage root=/dev/hda2 ro
```

Bennett, 3

Once the system was fairly stable, the 6MB partition was loaded into the RAM disk. This is very similar to how the RAM disk is loaded from EPROM, but development went faster since EPROMs weren't being programmed. To test the system without programming EPROMs, the system booted DOS and ran loadlin with the line:

```
loadlin zimage root=/dev/hda2 ramdisk=6144 ro
```

Because of the modification to ramdisk.c, the /dev/hda2 disk image is loaded into the RAM disk, then the root file system is switched to the RAM disk. The process of refining the disk image continues until everything is "perfect".

Bennett, 4

1.2 initializing a central processing unit of the computer system;	Bennett, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Bennett discloses this claim limitation:

System Operation

For booting, two options were considered:

- booting DOS, then running the loadlin program (to load Linux) from autoexec.bat
- installing LILO and booting Linux directly

The advantage of the second option would be a slightly shorter boot time. However, we implemented the first option, because we wanted to use a programmable keyboard—the software for programming the keyboard runs under DOS.

Bennett, 1

The boot sequence is:

- Power up and run memory tests
- load DOS which executes AUTOEXEC.BAT
  - run the keyboard programming application
  - run loadlin—this reads a linux kernel from the DOS disk & executes it
- the linux kernel takes over
- load the RAM disk from the EPROM disk
- switch the root file system to the RAM disk
- init will read inittab which executes dboot instead of getty
- the operator interface application is started

Bennett, 2

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Bennett, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Bennett discloses this claim limitation:

System Operation

For booting, two options were considered:

- booting DOS, then running the loadlin program (to load Linux) from autoexec.bat
- installing LILO and booting Linux directly

The advantage of the second option would be a slightly shorter boot time. However, we implemented the first option, because we wanted to use a programmable keyboard—the software for programming the keyboard runs under DOS.

Bennett, 1

When deciding how to implement the EPROM device driver, the first idea was to create an image of a disk in the EPROM. This would provide a RAM disk of the same size as the EPROM, 3.5MB in this case (the DOS portion of the SSD takes 1/2 MB). Instead, to allow a larger RAM disk, a compressed disk image is used. The compression used is simple—any sectors which are identical are only stored once. The primary advantage this gives is blank areas of the disk image don't need to take up EPROM space. Listing 1 shows the SSD disk compression used.

Bennett, 1

The boot sequence is:

- Power up and run memory tests
- load DOS which executes AUTOEXEC.BAT
  - run the keyboard programming application
  - run loadlin—this reads a linux kernel from the DOS disk & executes it
- the linux kernel takes over
- load the RAM disk from the EPROM disk
- switch the root file system to the RAM disk
- init will read inittab which executes dboot instead of getty
- the operator interface application is started

Bennett, 2

The first phase of disk image development was identifying the required and the desired items. The first step was to come up with a minimal system and then add the items required for the operator interface. Not being a Unix expert, coming up with the minimal system ended up being something of a trial and error process. I started with what I thought was needed, then tried running it. When an error occurred because of a missing program or library, that file was added. This process went on until the system ran happily.

The bulk of this was done by copying files from the "full" Linux partition to the 6MB partition, booting DOS and using the loadlin line:

```
loadlin zimage root=/dev/hda2 ro
```

Bennett, 3

Once the system was fairly stable, the 6MB partition was loaded into the RAM disk. This is very similar to how the RAM disk is loaded from EPROM, but development went faster since EPROMs weren't being programmed. To test the system without programming EPROMs, the system booted DOS and ran loadlin with the line:

```
loadlin zimage root=/dev/hda2 ramdisk=6144 ro
```

Because of the modification to ramdisk.c, the /dev/hda2 disk image is loaded into the RAM disk, then the root file system is switched to the RAM disk. The process of refining the disk image continues until everything is "perfect".

Bennett, 4

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Bennett, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Bennett discloses this claim limitation:

System Operation

For booting, two options were considered:

- booting DOS, then running the loadlin program (to load Linux) from autoexec.bat
- installing LILO and booting Linux directly

The advantage of the second option would be a slightly shorter boot time. However, we implemented the first option, because we wanted to use a programmable keyboard—the software for programming the keyboard runs under DOS.

Bennett, 1

When deciding how to implement the EPROM device driver, the first idea was to create an image of a disk in the EPROM. This would provide a RAM disk of the same size as the EPROM, 3.5MB in this case (the DOS portion of the SSD takes 1/2 MB). Instead, to allow a larger RAM disk, a compressed disk image is used. The compression used is simple—any sectors which are identical are only stored once. The primary advantage this gives is blank areas of the disk image don't need to take up EPROM space. Listing 1 shows the SSD disk compression used.

Appendix A26
Invalidity of U.S. Patent 7,181,608 based on Bennett

Bennett, 1

The boot sequence is:

- Power up and run memory tests
- load DOS which executes AUTOEXEC.BAT
  - run the keyboard programming application
  - run loadlin—this reads a linux kernel from the DOS disk & executes it
- the linux kernel takes over
- load the RAM disk from the EPROM disk
- switch the root file system to the RAM disk
- init will read inittab which executes dboot instead of getty
- the operator interface application is started

Bennett, 2

The first phase of disk image development was identifying the required and the desired items. The first step was to come up with a minimal system and then add the items required for the operator interface. Not being a Unix expert, coming up with the minimal system ended up being something of a trial and error process. I started with what I thought was needed, then tried running it. When an error occurred because of a missing program or library, that file was added. This process went on until the system ran happily.

The bulk of this was done by copying files from the "full" Linux partition to the 6MB partition, booting DOS and using the loadlin line:

```
loadlin zimage root=/dev/hda2 ro
```

Bennett, 3

Once the system was fairly stable, the 6MB partition was loaded into the RAM disk. This is very similar to how the RAM disk is loaded from EPROM, but development went faster since EPROMs weren't being programmed. To test the system without programming EPROMs, the system booted DOS and ran loadlin with the line:

```
loadlin zimage root=/dev/hda2 ramdisk=6144 ro
```

Because of the modification to ramdisk.c, the /dev/hda2 disk image is loaded into the RAM disk, then the root file system is switched to the RAM disk. The process of refining the disk image continues until everything is "perfect".

Bennett, 4

Bennett Claim 1.4

"servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Bennett, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Bennett discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

This is a quick look at making Linux bootable from EPROM on a 486 single board computer.

This article describes one way to run Linux in an embedded system with no hard disk. The application described is an Operator Interface in a monitor and display system developed by Boeing Flight Test. The airborne environment requires something fairly rugged which can withstand common power interruptions. To meet these requirements we decided to build the operator interface without a hard disk.

Bennett, 1

In order to automatically run the operator interface application, a program was written to replace getty. This program (dboot.c) will run login for a given user, and set the stdin, stdout and stderr to the specified virtual console.

The boot sequence is:

- Power up and run memory tests
- load DOS which executes AUTOEXEC.BAT
  - run the keyboard programming application
  - run loadlin—this reads a linux kernel from the DOS disk & executes it
- the linux kernel takes over
- load the RAM disk from the EPROM disk
- switch the root file system to the RAM disk
- init will read inittab which executes dboot instead of getty
- the operator interface application is started

Bennett, 2

Bennett Claim 2

"The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Bennett, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 1.3, and 1.4 above.	

4. The method of claim 1, further comprising updating the list of boot data.	Bennett, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Bennett, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Bennett, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Bennett, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Bennett, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 1.2, 1.3, and 1.4 above.	

7.2 a cache memory device; and;	Bennett, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Bennett, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Bennett, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Bennett, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Bennett, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Bennett discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

This is a quick look at making Linux bootable from EPROM on a 486 single board computer.

This article describes one way to run Linux in an embedded system with no hard disk. The application described is an Operator Interface in a monitor and display system developed by Boeing Flight Test. The airborne environment requires something fairly rugged which can withstand common power interruptions. To meet these requirements we decided to build the operator interface without a hard disk.

Bennett, 1

In order to automatically run the operator interface application, a program was written to replace getty. This program (dboot.c) will run login for a given user, and set the stdin, stdout and stderr to the specified virtual console.

The boot sequence is:

- Power up and run memory tests
- load DOS which executes AUTOEXEC.BAT
  - run the keyboard programming application
  - run loadlin—this reads a linux kernel from the DOS disk & executes it
- the linux kernel takes over
- load the RAM disk from the EPROM disk
- switch the root file system to the RAM disk
- init will read inittab which executes dboot instead of getty
- the operator interface application is started

Bennett, 2

9.1 maintaining a list of application data associated with an application program;	Bennett, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Bennett, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Bennett, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Bennett discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

This is a quick look at making Linux bootable from EPROM on a 486 single board computer.

This article describes one way to run Linux in an embedded system with no hard disk. The application described is an Operator Interface in a monitor and display system developed by Boeing Flight Test. The airborne environment requires something fairly rugged which can withstand common power interruptions. To meet these requirements we decided to build the operator interface without a hard disk.

Bennett, 1

In order to automatically run the operator interface application, a program was written to replace getty. This program (dboot.c) will run login for a given user, and set the stdin, stdout and stderr to the specified virtual console.

The boot sequence is:

- Power up and run memory tests
- load DOS which executes AUTOEXEC.BAT
  - run the keyboard programming application
  - run loadlin—this reads a linux kernel from the DOS disk & executes it
- the linux kernel takes over
- load the RAM disk from the EPROM disk
- switch the root file system to the RAM disk
- init will read inittab which executes dboot instead of getty
- the operator interface application is started

Bennett, 2

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Bennett, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Bennett discloses this limitation:

Bennett discloses this claim limitation:

System Operation

For booting, two options were considered:

- booting DOS, then running the loadlin program (to load Linux) from autoexec.bat
- installing LILO and booting Linux directly

The advantage of the second option would be a slightly shorter boot time. However, we implemented the first option, because we wanted to use a programmable keyboard—the software for programming the keyboard runs under DOS.

Bennett, 1

When deciding how to implement the EPROM device driver, the first idea was to create an image of a disk in the EPROM. This would provide a RAM disk of the same size as the EPROM, 3.5MB in this case (the DOS portion of the SSD takes 1/2 MB). Instead, to allow a larger RAM disk, a compressed disk image is used. The compression used is simple—any sectors which are identical are only stored once. The primary advantage this gives is blank areas of the disk image don't need to take up EPROM space. Listing 1 shows the SSD disk compression used.

Bennett, 1

The boot sequence is:

- Power up and run memory tests
- load DOS which executes AUTOEXEC.BAT
  - run the keyboard programming application
  - run loadlin—this reads a linux kernel from the DOS disk & executes it
- the linux kernel takes over
- load the RAM disk from the EPROM disk
- switch the root file system to the RAM disk
- init will read inittab which executes dboot instead of getty
- the operator interface application is started

Bennett, 2

The first phase of disk image development was identifying the required and the desired items. The first step was to come up with a minimal system and then add the items required for the operator interface. Not being a Unix expert, coming up with the minimal system ended up being something of a trial and error process. I started with what I thought was needed, then tried running it. When an error occurred because of a missing program or library, that file was added. This process went on until the system ran happily.

The bulk of this was done by copying files from the "full" Linux partition to the 6MB partition, booting DOS and using the loadlin line:

```
loadlin zimage root=/dev/hda2 ro
```

Bennett, 3

Once the system was fairly stable, the 6MB partition was loaded into the RAM disk. This is very similar to how the RAM disk is loaded from EPROM, but development went faster since EPROMs weren't being programmed. To test the system without programming EPROMs, the system booted DOS and ran loadlin with the line:

```
loadlin zimage root=/dev/hda2 ramdisk=6144 ro
```

Because of the modification to ramdisk.c, the /dev/hda2 disk image is loaded into the RAM disk, then the root file system is switched to the RAM disk. The process of refining the disk image continues until everything is "perfect".

Bennett, 4

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Bennett, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Bennett discloses this limitation:

Bennett discloses this claim limitation:

System Operation

For booting, two options were considered:

- booting DOS, then running the loadlin program (to load Linux) from autoexec.bat
- installing LILO and booting Linux directly

The advantage of the second option would be a slightly shorter boot time. However, we implemented the first option, because we wanted to use a programmable keyboard—the software for programming the keyboard runs under DOS.

Bennett, 1

When deciding how to implement the EPROM device driver, the first idea was to create an image of a disk in the EPROM. This would provide a RAM disk of the same size as the EPROM, 3.5MB in this case (the DOS portion of the SSD takes 1/2 MB). Instead, to allow a larger RAM disk, a compressed disk image is used. The compression used is simple—any sectors which are identical are only stored once. The primary advantage this gives is blank areas of the disk image don't need to take up EPROM space. Listing 1 shows the SSD disk compression used.

Bennett, 1

The boot sequence is:

- Power up and run memory tests
- load DOS which executes AUTOEXEC.BAT
  - run the keyboard programming application
  - run loadlin—this reads a linux kernel from the DOS disk & executes it
- the linux kernel takes over
- load the RAM disk from the EPROM disk
- switch the root file system to the RAM disk
- init will read inittab which executes dboot instead of getty
- the operator interface application is started

Bennett, 2

The first phase of disk image development was identifying the required and the desired items. The first step was to come up with a minimal system and then add the items required for the operator interface. Not being a Unix expert, coming up with the minimal system ended up being something of a trial and error process. I started with what I thought was needed, then tried running it. When an error occurred because of a missing program or library, that file was added. This process went on until the system ran happily.

The bulk of this was done by copying files from the "full" Linux partition to the 6MB partition, booting DOS and using the loadlin line:

```
loadlin zimage root=/dev/hda2 ro
```

Bennett, 3

Once the system was fairly stable, the 6MB partition was loaded into the RAM disk. This is very similar to how the RAM disk is loaded from EPROM, but development went faster since EPROMs weren't being programmed. To test the system without programming EPROMs, the system booted DOS and ran loadlin with the line:

```
loadlin zimage root=/dev/hda2 ramdisk=6144 ro
```

Because of the modification to ramdisk.c, the /dev/hda2 disk image is loaded into the RAM disk, then the root file system is switched to the RAM disk. The process of refining the disk image continues until everything is "perfect".

Bennett, 4

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Bennett, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Bennett, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Bennett, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Bennett discloses this limitation:

See Claims 1.3 and 1.4 above.

See also

When deciding how to implement the EPROM device driver, the first idea was to create an image of a disk in the EPROM. This would provide a RAM disk of the same size as the EPROM, 3.5MB in this case (the DOS portion of the SSD takes 1/2 MB). Instead, to allow a larger RAM disk, a compressed disk image is used. The compression used is simple—any sectors which are identical are only stored once. The primary advantage this gives is blank areas of the disk image don't need to take up EPROM space. Listing 1 shows the SSD disk compression used.

EPROM Disk Compression
(/files/linuxjournal.com/linuxjournal/articles/002/0243/0243f1.jpg)

Bennett, 1

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Bennett, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Bennett, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Bennett, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

22.1 maintaining a list of boot data used for booting a computer system;.	Bennett, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Bennett, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Bennett, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Bennett, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Bennett, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Bennett, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Bennett, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

27.1 a boot device..	Bennett, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Bennett, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Bennett, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,	Bennett, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Bennett, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Bennett, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Bennett, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Bennett, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Bennett, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Bennett discloses this limitation: *See* Claims 15 and 27 above	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Bennett, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

Appendix A27
Invalidity of U.S. Patent 7,181,608 based on Burrows

The publication Michael Burrows, Charles Jerian, Butler Lampson, and Timothy Mann, On-line data compression in a log-structured file System("Burrows") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A27
Invalidity of U.S. Patent 7,181,608 based on Burrows

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Burrows, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.1 maintaining a list of boot data used for booting a computer system;	Burrows, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.2 initializing a central processing unit of the computer system;	Burrows, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Burrows, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Burrows discloses this limitation:

> "We have incorporated on-line data compression into the low levels of a log-structured file system (Rosenblum's Sprite LFS). Each block of data or meta-data is compressed as it is written to the disk and decompressed as it is read."

Burrows, Abstract.

> "The module that reads a disk block given its logical address needs a way to find the physical address of the compressed bytes. We keep a logical block map for each segment, which is simply an array indexed by compression block number, whose entries are the physical byte addresses of the blocks relative to the start of the segment. The block map is constructed in memory as the segment is being compressed, and written to the end of the segment when the segment is full. The maps are needed for all file reads, so they are cached in memory whenever possible."

Burrows, 5.

> "The procedure for finding the compressed data associated with a logical address is as follows:
>
> > 1. Extract the segment number from the logical address. Use it to find the logical block map for the segment.
> >
> > 2. Extract the compression block number from the address. Use it

to index the logical block map. This yields the physical byte offset of the compressed data within the segment.

3. Examine the next entry in the map, to find the start of the next block. This determines how much data should be read from the disk.

4. Read the compressed data from the disk and decompress it.

5. Extract the sector number from the logical address. Use it to identify the desired sector within the decompressed block."

Burrows, 5-6.

"Unfortunately, this procedure reads and decompresses a full compression block even if the caller wanted only some smaller unit, such as a file system block or a physical sector. We alleviate this problem by caching the entire decompressed block in memory, rather than just caching the requested sectors. The data could be placed in the file system buffer cache, but for simplicity in our prototype, we cached the last decompressed block within the read routine. Sprite LFS reads files sequentially in 4 KByte units, so this simple caching strategy typically achieves three hits for each 16 KByte compression block when reading large files.

When the file system is reading non-sequentially, the additional time to read a full compression block cannot be hidden by caching. Fortunately, this time is small compared to the rotational latency. The time needed to decompress the full block in software is several milliseconds; it would be much smaller if decompression were implemented in hardware."

Burrows, 6.

"Conceptually, it is simple to write compressed data to the log. As LFS constructs its element list, it can compress the data into a buffer, one compression block at a time."

Burrows, 6.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Burrows, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Burrows discloses this limitation:

> "We have incorporated on-line data compression into the low levels of a log-structured file system (Rosenblum's Sprite LFS). Each block of data or meta-data is compressed as it is written to the disk and decompressed as it is read."

Burrows, Abstract.

> "Hardware compression devices mesh well with this design. Chips are already available that operate at speeds exceeding disk transfer rates, which indicates that hardware compression would not only remove the performance degradation we observed, but might well increase the effective disk transfer rate beyond that obtainable from a system without compression."

Burrows, Abstract.

> "Building a file system that compresses the data it stores on disk is clearly an attractive idea. First, more data would fit on the disk. Also, if a fast

hardware data compressor could be put into the data path to disk, it would increase the effective disk transfer rate by the compression factor, thus speeding up the system."

Burrows, 1.

"The module that reads a disk block given its logical address needs a way to find the physical address of the compressed bytes. We keep a logical block map for each segment, which is simply an array indexed by compression block number, whose entries are the physical byte addresses of the blocks relative to the start of the segment. The block map is constructed in memory as the segment is being compressed, and written to the end of the segment when the segment is full. The maps are needed for all file reads, so they are cached in memory whenever possible."

Burrows, 5.

"The procedure for finding the compressed data associated with a logical address is as follows:

1. Extract the segment number from the logical address. Use it to find the logical block map for the segment.

2. Extract the compression block number from the address. Use it to index the logical block map. This yields the physical byte offset of the compressed data within the segment.

3. Examine the next entry in the map, to find the start of the next block. This determines how much data should be read from the disk.

4. Read the compressed data from the disk and decompress it.

5. Extract the sector number from the logical address. Use it to identify the desired sector within the decompressed block."

Burrows, 5-6.

"Unfortunately, this procedure reads and decompresses a full compression block even if the caller wanted only some smaller unit, such as a file system block or a physical sector. We alleviate this problem by caching the entire decompressed block in memory, rather than just caching the requested sectors. The data could be placed in the file system buffer cache, but for simplicity in our prototype, we cached the last

> decompressed block within the read routine. Sprite LFS reads files sequentially in 4 KByte units, so this simple caching strategy typically achieves three hits for each 16 KByte compression block when reading large files.
>
> When the file system is reading non-sequentially, the additional time to read a full compression block cannot be hidden by caching. Fortunately, this time is small compared to the rotational latency. The time needed to decompress the full block in software is several milliseconds; it would be much smaller if decompression were implemented in hardware."

Burrows, 6.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Burrows, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Burrows discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "A second issue is that applications often commit small amounts of data to disk, resulting in poor compression."

Burrows, 9.

> "The scheme benefits from concentrating on executable files, which are read by few things besides the operating system itself; no attempt is made to make the compression transparent to other applications."

Burrows, 18.

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Burrows, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Burrows discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

> "These problems suggest that the compressor should be placed in the disk controller, but doing so requires more changes to LFS, and quite specialized hardware."

Burrows, 12.

> "But it is not possible to tell whether a particular disk block will fit in the current segment until compression has taken place, so we cannot determine what data should be written until the data has been transferred to the controller for compression."

Burrows, 12-13.

> "Once forward references are eliminated, our problem can be solved by placing a large buffer in the disk controller to hold the compressed data."

Burrows, 13.

> "Thus, the file system can prepare a larger amount of data for writing than will actually fit in the current segment, and can instruct the controller to truncate the write if it occupies more than the amount of physical space available."

Burrows, 13.

> "Finally, a disk controller containing a compressor must inform the software where each compressed block fell on the disk. Ideally, it would also construct the logical block map and append it to the data being written, in order to avoid an extra disk transfer to place the map at the end of the segment."

Burrows, 13.

> "The design can be adapted to use hardware compression devices, either combined with a disk controller or packaged separately."

Burrows, 18.

4. The method of claim 1, further comprising updating the list of boot data.	Burrows, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claim 1.1 above.	

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Burrows, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Burrows, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Burrows, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Burrows, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Burrows discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

> "These problems suggest that the compressor should be placed in the disk controller, but doing so requires more changes to LFS, and quite specialized hardware."

Burrows, 12.

> "But it is not possible to tell whether a particular disk block will fit in the current segment until compression has taken place, so we cannot determine what data should be written until the data has been transferred to the controller for compression."

Burrows, 12-13.

> "Once forward references are eliminated, our problem can be solved by placing a large buffer in the disk controller to hold the compressed data."

Burrows, 13.

> "Thus, the file system can prepare a larger amount of data for writing than will actually fit in the current segment, and can instruct the controller to truncate the write if it occupies more than the amount of physical space available."

Burrows, 13.

> "Finally, a disk controller containing a compressor must inform the software where each compressed block fell on the disk. Ideally, it would

> also construct the logical block map and append it to the data being written, in order to avoid an extra disk transfer to place the map at the end of the segment."
>
> Burrows, 13.
>
> "The design can be adapted to use hardware compression devices, either combined with a disk controller or packaged separately."
>
> Burrows, 18.

7.2 a cache memory device; and;	Burrows, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Burrows, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Burrows, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Burrows, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Burrows, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Burrows discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> "A second issue is that applications often commit small amounts of data to disk, resulting in poor compression."

Burrows, 9.

> "The scheme benefits from concentrating on executable files, which are read by few things besides the operating system itself; no attempt is made to make the compression transparent to other applications."

Burrows, 18.

9.1 maintaining a list of application data associated with an application program;	Burrows, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Burrows, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Burrows, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Burrows discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> "A second issue is that applications often commit small amounts of data to disk, resulting in poor compression."

Burrows, 9.

> "The scheme benefits from concentrating on executable files, which are read by few things besides the operating system itself; no attempt is made to make the compression transparent to other applications."

Burrows, 18.

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Burrows, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Burrows discloses this limitation:

> "We have incorporated on-line data compression into the low levels of a log-structured file system (Rosenblum's Sprite LFS). Each block of data or meta-data is compressed as it is written to the disk and decompressed as it is read."

Burrows, Abstract.

> "The module that reads a disk block given its logical address needs a way to find the physical address of the compressed bytes. We keep a logical block map for each segment, which is simply an array indexed by compression block number, whose entries are the physical byte addresses of the blocks relative to the start of the segment. The block map is constructed in memory as the segment is being compressed, and written to the end of the segment when the segment is full. The maps are needed for all file reads, so they are cached in memory whenever possible."

Burrows, 5.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Burrows, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Burrows discloses this limitation:

> "We have incorporated on-line data compression into the low levels of a log-structured file system (Rosenblum's Sprite LFS). Each block of data or meta-data is compressed as it is written to the disk and decompressed as it is read."

Burrows, Abstract.

> "The procedure for finding the compressed data associated with a logical address is as follows:
>
> > 1. Extract the segment number from the logical address. Use it to find the logical block map for the segment.
> >
> > 2. Extract the compression block number from the address. Use it to index the logical block map. This yields the physical byte offset of the compressed data within the segment.
> >
> > 3. Examine the next entry in the map, to find the start of the next block. This determines how much data should be read from the disk.
> >
> > 4. Read the compressed data from the disk and decompress it.
> >
> > 5. Extract the sector number from the logical address. Use it to identify the desired sector within the decompressed block."

Burrows, 5-6.

> "Unfortunately, this procedure reads and decompresses a full compression block even if the caller wanted only some smaller unit, such as a file system block or a physical sector. We alleviate this problem by caching the entire decompressed block in memory, rather than just

caching the requested sectors. The data could be placed in the file system buffer cache, but for simplicity in our prototype, we cached the last decompressed block within the read routine. Sprite LFS reads files sequentially in 4 KByte units, so this simple caching strategy typically achieves three hits for each 16 KByte compression block when reading large files.

When the file system is reading non-sequentially, the additional time to read a full compression block cannot be hidden by caching. Fortunately, this time is small compared to the rotational latency. The time needed to decompress the full block in software is several milliseconds; it would be much smaller if decompression were implemented in hardware."

Burrows, 6.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Burrows, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Burrows, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Burrows, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Burrows discloses this limitation:

See Claims 1.3 and 1.4 above.

> "Besides the algorithms based on Wheeler's ideas, we tried the LZRW1-A and LZRW3-A algorithms due to Williams. A full description and implementation are available elsewhere [15, 16], so we omit the details here."

Burrows, 11. *See also* Table 2.

> "For comparison, we include figures for the popular compress utility, which uses the LZC algorithm, and the zoo archiver, which uses the LZSS algorithm. The table shows that the algorithms we chose are quite fast, but better compression could be obtained by sacrificing speed.
>
> Bell, Witten, and Cleary give a more thorough comparison of compression algorithms and their effectiveness on different sorts of data [3]. They also describe the LZC and LZSS algorithms."

Burrows, 11-12.

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Burrows, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.3, 1.4, and 15 above.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Burrows, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Burrows, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above.	

22.1 maintaining a list of boot data used for booting a computer system;.	Burrows, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claim 1.1 above.	

22.2 initializing a central processing unit of the computer system;.	Burrows, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claim 1.2 above.	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Burrows, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claim 1.3 above.	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Burrows, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claim 1.4 above.	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Burrows, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 4, 10 and 11 above.	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Burrows, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 15 and 22 above.	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Burrows, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 16 and 22 above.	

27.1 a boot device..	Burrows, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claim 1 above.	

27.2 a processor..	Burrows, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claim 1.2 above.	

27.3 cache memory; and.	Burrows, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.3 and 1.4 above.	

27.4 non-volatile memory for storing logic code for use by the processor,..	Burrows, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claim 1.1 and 1.3 above.	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Burrows, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claim 1.1 above.	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Burrows, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claim 1.3 above.	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Burrows, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claim 1.4 above.	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Burrows, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 4, 10, and 11 above.	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Burrows, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 15 and 27 above.	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Burrows, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 16 and 27 above.	

Appendix A28
Invalidity of U.S. Patent 7,181,608 based on Cheng

The publication Cheng et al., Fast and highly reliable IBMLZ1 compression chip and algorithm for storage, Hot Chips V11, August 1995 ("Cheng") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A28
Invalidity of U.S. Patent 7,181,608 based on Cheng

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Cheng, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Cheng discloses this limitation:

> "Early Objectives: 16X speed-up, 80% cost reduction"

Cheng, 144.

> "Compression Benefits for Storage System
>
> • Compression x Compaction = 4.5 X - 6.0 X
>
> •System Performance"

Cheng, 144. *See also* Cheng, 144 Fig.1, Fig. 2.

> "Data compression allows more efficient use of storage media and communication bandwidth. Standard compression offerings for tape storage have been well-established since the late 1980s."

Cheng, 155.

> "The desire to extend these cost/performance benefits to higher data-rate media and broader media forms, such as DASD storage subsystems, motivated the design and development of the IBMLZ1 compression algorithm and technology."

Cheng, 155.

> "The addition of compression capability to the DASD subsystem facilitates more efficient use of subsystem resources: cache, data path bandwidth, and disk capacity in a transparent manner to the system storing the data."

Cheng, 155.

> "Compression allows existing platforms and applications to benefit from a lower storage cost and potentially higher performance without any change to system hardware or software."

Cheng, 155.

> "If the data is compressed as it enters the subsystem, see Fig. 1, the cache resource has effectively been tripled. Customers can benefit either from improved performance due to better hit ratios, or reduce their cost by configuring smaller amounts of cache. An added benefit of data compression upon entry to the subsystem is the reduced utilization of internal buses and DASD paths as data flows through the subsystem. Finally, sequential performance can be improved. In general, on high end systems with ESCON attached DASD, the throughput of sequential operations is gated by the DASD data rate, typically less than the 18 MB/sec capability of the channel. When the device is transferring compressed data, the transfer rate is effectively multiplied by th compression ratio, allowing the full capability of the channel to be realized."

Cheng, 155. *See also* Fig. 1.

> "Finally, sequential performance can be improved. In general, on high end systems with ESCON attached DASD, the throughput of sequential operations is gated by the DASD data rate, typically less than the 18 MB/sec capability of the channel. When the device is transferring compressed data, the transfer rate is effectively multiplied by the compression ratio, allowing the full capability of the channel to be realized."

Cheng, 155.

> "The IBMLZl algorithm and technology was designed for high compression/decompression throughput with efficient hardware implementation, high reliability, low system overhead, and robust compression. Data integrity and reliability are ensured by coupled compression-decompression checking, scrubbing operation, and extensive build-in checkings. Extremely low CPB=l^6 compression and decompression have been achieved. The extremely high compressing/decompressing throughput of 30 MB/sec-50 MB/sec allows transparent mode of operation and hence achieved minimal system overhead. The IBMLZl algorithm compresses well over the VM, MVS,

RS6000, and PC test cases." Cheng, 163.

1.1 maintaining a list of boot data used for booting a computer system;	Cheng, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.2 initializing a central processing unit of the computer system;	Cheng, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Cheng, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Cheng, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Cheng discloses this limitation:

> "Data compression allows more efficient use of storage media and communication bandwidth. Standard compression offerings for tape storage have been well-established since the late 1980s."

Cheng, 155.

> "The addition of compression capability to the DASD subsystem facilitates more efficient use of subsystem resources: cache, data path bandwidth, and disk capacity in a transparent manner to the system storing the data."

Cheng, 155.

> "Compression allows existing platforms and applications to benefit from a lower storage cost and potentially higher performance without any change to system hardware or software."

Cheng, 155.

> "If the data is compressed as it enters the subsystem, see Fig. 1, the cache resource has effectively been tripled. Customers can benefit either from improved performance due to better hit ratios, or reduce their cost by configuring smaller amounts of cache. An added benefit of data compression upon entry to the subsystem is the reduced utilization of internal buses and DASD paths as data flows through the subsystem. Finally, sequential performance can be improved. In general, on high end systems with ESCON attached DASD, the throughput of sequential operations is gated by the DASD data rate, typically less than the 18 MB/sec capability of the channel. When the device is transferring compressed data, the transfer rate is effectively multiplied by th compression ratio, allowing the full capability of the channel to be realized."

Cheng, 155. *See also* Fig. 1.

> "Finally, sequential performance can be improved. In general, on high end systems with ESCON attached DASD, the throughput of sequential operations is gated by the DASD data rate, typically less than the 18 MB/sec capability of the channel. When the device is transferring compressed data, the transfer rate is effectively multiplied by the compression ratio, allowing the full capability of the channel to be realized."

Cheng, 155.

> "The IBMLZl algorithm and technology was designed for high compression/decompression throughput with efficient hardware implementation, high reliability, low system overhead, and robust compression. Data integrity and reliability are ensured by coupled compression-decompression checking, scrubbing operation, and extensive build-in checkings. Extremely low CPB=l[6] compression and decompression have been achieved. The extremely high compressing/decompressing throughput of 30 MB/sec-50 MB/sec allows transparent mode of operation and hence achieved minimal system overhead. The IBMLZl algorithm compresses well over the VM, MVS, RS6000, and PC test cases."

Cheng, 163.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Cheng, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Cheng discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "The compression technology lowers the cost of storage without changes to any applications or data access methods."

Cheng, 155.

> "Compression allows existing platforms and applications to benefit from a lower storage cost and potentially higher performance without any change to system hardware or software."

Cheng, 155.

> "Robust Compression: achieve good coding efficiency for broad applications."

Cheng, 159.

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Cheng, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Cheng discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

"DASO (DISK) Controller Compression and IBMLZ"

Cheng, 144.

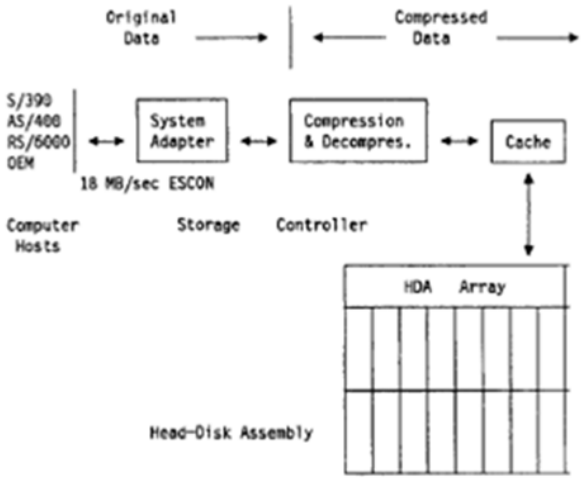


Fig. 1.

"Compression Objectives for Storage Controller"

Cheng, 144. *See also generally*, Cheng 144-145 (Compression Objectives for Storage Controller).

> "The IBMLZl is used in IBM's high-performance DASD controller family, tape drive family, and the AIX file system with compression."

Cheng, 160.

> "IBMLZl Compression Technology for Storage Controller"

Cheng, 163. *See also generally*, Cheng, 163 (IBMLZ1 Compression Technology for Storage Controller).

4. The method of claim 1, further comprising updating the list of boot data.	Cheng, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claim 1.1 above.	

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Cheng, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Cheng, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Cheng, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Cheng, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Cheng discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

"DASO (DISK) Controller Compression and IBMLZ"

Cheng, 144.

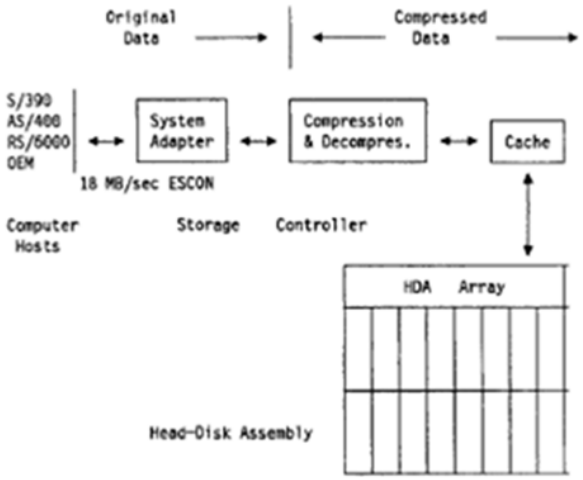


Fig. 1.

"Compression Objectives for Storage Controller"

Cheng, 144. *See also generally*, Cheng 144-145 (Compression Objectives for Storage Controller).

"The IBMLZl is used in IBM's high-performance DASD controller family, tape drive family, and the AIX file system with compression."

Cheng, 160.

"IBMLZl Compression Technology for Storage Controller"

Cheng, 163. *See also generally*, Cheng, 163 (IBMLZ1 Compression Technology for Storage Controller).

7.2 a cache memory device; and;	Cheng, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Cheng, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Cheng, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Cheng, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Cheng, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Cheng discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> "The compression technology lowers the cost of storage without changes to any applications or data access methods."

Cheng, 155.

> "Compression allows existing platforms and applications to benefit from a lower storage cost and potentially higher performance without any change to system hardware or software."

Cheng, 155.

> "Robust Compression: achieve good coding efficiency for broad applications."

Appendix A28
Invalidity of U.S. Patent 7,181,608 based on Cheng

Cheng, 159.

9.1 maintaining a list of application data associated with an application program;	Cheng, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Cheng, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Cheng, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Cheng discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> "The compression technology lowers the cost of storage without changes to any applications or data access methods."

Cheng, 155.

> "Compression allows existing platforms and applications to benefit from a lower storage cost and potentially higher performance without any change to system hardware or software."

Cheng, 155.

> "Robust Compression: achieve good coding efficiency for broad applications."

Cheng, 159.

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Cheng, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Cheng discloses this limitation:

See ***Copy from 936 element 1.1***

> "Compression Benefits for Storage System
>
> • Compression x Compaction = 4.5 X - 6.0 X
>
> •System Performance"

Cheng, 144. *See also* Cheng, 144 Fig.1, Fig. 2.

> "Data compression allows more efficient use of storage media and communication bandwidth. Standard compression offerings for tape storage have been well-established since the late 1980s."

Cheng, 155.

> "Compression allows existing platforms and applications to benefit from a lower storage cost and potentially higher performance without any change to system hardware or software."

Cheng, 155.

> "The IBMLZl algorithm and technology was designed for high compression/decompression throughput with efficient hardware implementation, high reliability, low system overhead, and robust compression. Data integrity and reliability are ensured by coupled compression-decompression checking, scrubbing operation, and

> extensive build-in checkings. Extremely low CPB=l^6 compression and decompression have been achieved. The extremely high compressing/decompressing throughput of 30 MB/sec-50 MB/sec allows transparent mode of operation and hence achieved minimal system overhead. The IBMLZl algorithm compresses well over the VM, MVS, RS6000, and PC test cases."
>
> Cheng, 163.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Cheng, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Cheng, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Cheng, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Cheng, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Cheng discloses this limitation:

See Claims 1.3 and 1.4 above.

See also

"DASO (DISK) Controller Compression and IBMLZ23

•Lossless and General-Purpose Compression

•Simple Algorithm Format and Robust

• Optimized for High-Thruput Hardware Execution: 1 Byte/Cycle,

40MH/sec.

• Algorithm Accepted as QIC 154 Standard"

Cheng, 144.

"IBMLZ1 Algorithm and Hardware Development Paradigm

Extremely High Reliability One undetected Error in 1030

Efficiencies Robust Compression, Speed, Latency

System Requirement Intra-Parallel instead Inter-Parallel, good

Compression, Speed

Silicon Technology Regularity"

Cheng, 146. *See also generally*, Cheng, 147-154.

> "The IBMLZ1 compression algorithm was designed not only for robust and highly efficient compression, but also for extremely high reliability. As compression removes redundancy in the source, the compressed data becomes extremely vulnerable to data corruption. Key design objectives for the IBMLZ1 development were: efficient hardware execution and efficient use of silicon technology; and minimum system integration overhead. Through new observations of pattern matching match-length distribution and use of graph vertex coloring for evaluating data flows, the IBMLZ1 compression algorithm and technology achieved the above objectives."

Cheng, 155.

> "The LZl and the LZ2 compression algorithms are commonly regarded as Lempel and Ziv's compression algorithm 1 [ZL 77] and algorithm 2 [ZL 78] . The LZl and the LZ2, in their original form, expressed the notion of coding Model and bounds on compression. Professor Lempel noted that more than 90 percent of the compression software in the PC world is derived from either LZl or LZ2 class algorithms."

Cheng, 157. *See also generally*, Cheng, 157-165.

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Cheng, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Cheng discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

See also

> "Common methods used for redundancy reduction are run-length encoding, pattern matching, transformation, transform coding, and so on."

Cheng, 156.

> "The Huffman code, however, remains as the most popular encoding method for its simplicity and its effectiveness in general."

Cheng, 156. *See also generally*, Cheng, 156-165, Fig. 6.

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Cheng, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Cheng, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above.	

22.1 maintaining a list of boot data used for booting a computer system;.	Cheng, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claim 1.1 above.	

22.2 initializing a central processing unit of the computer system;.	Cheng, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claim 1.2 above.	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Cheng, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claim 1.3 above.	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Cheng, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claim 1.4 above.	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Cheng, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 4, 10 and 11 above.	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Cheng, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 15 and 22 above.	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Cheng, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 16 and 22 above.	

27.1 a boot device..	Cheng, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claim 1 above.	

27.2 a processor..	Cheng, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claim 1.2 above.	

27.3 cache memory; and.	Cheng, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.3 and 1.4 above.	

27.4 non-volatile memory for storing logic code for use by the processor,..	Cheng, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claim 1.1 and 1.3 above.	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Cheng, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claim 1.1 above.	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Cheng, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claim 1.3 above.	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Cheng, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claim 1.4 above.	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Cheng, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 4, 10, and 11 above.	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Cheng, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 15 and 27 above.	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Cheng, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 16 and 27 above.	

Appendix A29
Invalidity of U.S. Patent 7,181,608 based on Craft

The publication Craft, A Fast hardware data compression algorithm and some algorithmic extension, IBM J. Res. Develop., Vol 43, Nov. 1998, ("Craft") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A29
Invalidity of U.S. Patent 7,181,608 based on Craft

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Craft, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Craft discloses this limitation:

> "This paper reports on work at IBM's Austin and Burlington laboratories concerning fast hardware implementations of general-purpose lossless data compression algorithms, particularly for use in enhancing the data capacity of computer storage devices or systems, and transmission data rates for networking or telecommunications channels"

Craft, 733.

> "The scope of the work quickly expanded, however, as it became apparent that data compression technology could have tremendous implications for some major IBM business segments. In particular, its deployment within computer systems to enhance DASD storage capacity, or to increase the effective bandwidth of networking data channels, would present a major competitive advantage."

Craft, 734.

1.1 maintaining a list of boot data used for booting a computer system;	Craft, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.2 initializing a central processing unit of the computer system;	Craft, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Craft, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Craft, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Craft discloses this limitation:

> "This paper reports on work at IBM's Austin and Burlington laboratories concerning fast hardware implementations of general-purpose lossless data compression algorithms, particularly for use in enhancing the data capacity of computer storage devices or systems, and transmission data rates for networking or telecommunications channels"

Craft, 733.

> "The scope of the work quickly expanded, however, as it became apparent that data compression technology could have tremendous implications for some major IBM business segments. In particular, its deployment within computer systems to enhance DASD storage capacity, or to increase the effective bandwidth of networking data channels, would present a major competitive advantage."

Craft, 734.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Craft, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Craft discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "It not only shows better compression on the smaller data block sizes that are desirable for random-access applications, it is also particularly amenable to a fast and simple CMOS hardware implementation based on the use of a content-addressable memory (CAM) array."

Craft, 734.

> "One cLDC preprocessor recodes runs of identical data bytes, and this is followed by one designed to recode the Unicode format, an increasingly important text data coding standard used by Java** and other Internet-based applications. ALDC compresses Unicode versions of ASCII text files some 40% worse than the original ASCII, but cLDC is able to remove this penalty completely."

Craft, 735.

See also generally, Craft 735-736 (Algorithms for DASD or networking applications).

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Craft, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.3, and 1.4 above.	

4. The method of claim 1, further comprising updating the list of boot data.	Craft, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claim 1.1 above.	

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Craft, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Craft, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Craft, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Craft, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.2, 1.3, and 1.4 above.	

7.2 a cache memory device; and;	Craft, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Craft, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Craft, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Craft, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Craft, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Craft discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> "It not only shows better compression on the smaller data block sizes that are desirable for random-access applications, it is also particularly amenable to a fast and simple CMOS hardware implementation based on the use of a content-addressable memory (CAM) array."

Craft, 734.

> "One cLDC preprocessor recodes runs of identical data bytes, and this is followed by one designed to recode the Unicode format, an increasingly important text data coding standard used by Java** and other Internet-based applications. ALDC compresses Unicode versions of ASCII text files some 40% worse than the original ASCII, but cLDC is able to remove this penalty completely."

Craft, 735. *See also generally*, Craft 735-736 (Algorithms for DASD or networking applications).

9.1 maintaining a list of application data associated with an application program;	Craft, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Craft, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Craft, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Craft discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> "It not only shows better compression on the smaller data block sizes that are desirable for random-access applications, it is also particularly amenable to a fast and simple CMOS hardware implementation based on the use of a content-addressable memory (CAM) array."

Craft, 734.

> "One cLDC preprocessor recodes runs of identical data bytes, and this is followed by one designed to recode the Unicode format, an increasingly important text data coding standard used by Java** and other Internet-based applications. ALDC compresses Unicode versions of ASCII text files some 40% worse than the original ASCII, but cLDC is able to remove this penalty completely."

Craft, 735.

Appendix A29
Invalidity of U.S. Patent 7,181,608 based on Craft

See also generally, Craft 735-736 (Algorithms for DASD or networking applications).

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Craft, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Craft discloses this limitation:

See ***Copy from 936 element 1.1***

> "This paper reports on work at IBM's Austin and Burlington laboratories concerning fast hardware implementations of general-purpose lossless data compression algorithms, particularly for use in enhancing the data capacity of computer storage devices or systems, and transmission data rates for networking or telecommunications channels"

Craft, 733.

> "The scope of the work quickly expanded, however, as it became apparent that data compression technology could have tremendous implications for some major IBM business segments. In particular, its deployment within computer systems to enhance DASD storage capacity, or to increase the effective bandwidth of networking data channels, would present a major competitive advantage."

Craft, 734.

> "The encoding format for ALDC is presented, together with details of IBM's current fast hardware CMOS compression engine designs, based on use of a content addressable memory (CAM) array."

Craft, 733.

> "The initial IBM ALDC chips used entirely separate decompression and compression engines [5] and were in fact configured so that both could

be operated simultaneously. However, in our later designs, we chose to add a conventional address decoder to the CAM so that it can also function as an SRAM during an LZ1 decode function. This results in a very compact hardware encoder/decoder, which we call a CRAM design, as it combines a compression engine CAM and a decompression engine RAM into one silicon array."

Craft, 738.

"For the few customers requiring simultaneous compression/decompression capability, we can therefore put two such CRAM engines on the same chip, and the silicon area needed is still modest."

Craft, 738.

"The hardware to implement both pre/postprocessors is trivial, requiring less than 10% of the CMOS chip area of the ALDC CRAM engine itself."

Craft, 744.

"Using the more advanced IBM CMOS 6 and CMOS 7 technologies, we can easily fit multiple engines onto a single chip. In turn, this allows us to design ALDC compression systems which have sustained throughput in the gigabyte-per-second range, if required, and should effectively meet system storage and networking application requirements into the next millennium."

Craft, 744.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Craft, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Craft, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Craft, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Craft, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Craft discloses this limitation:

See Claims 1.3 and 1.4 above.

See also

> "Then, two main classes of adaptive Lempel-Ziv algorithm, now known as LZ1 and LZ2, are introduced. An outline of early work comparing these two types of algorithm is presented, together with some fundamental distinctions which led to the choice and development of an IBM variant of the LZ1 algorithm, ALDC, and its implementation in hardware."

Craft, 733.

> "Adaptive data compression techniques try to construct models, or look for data sequences derived in some fashion from recent experience. The algorithm thus adapts dynamically to different types of data. There are two classes of adaptive algorithm which are generally acknowledged to be among the most effective, yielding good compression over a wide range of data types. These were both first proposed by A. Lempel and J. Ziv, in 1977 and 1978, and are commonly now referred to as LZ1 and LZ2 respectively [1-3].

Craft, 734. See also generally, Craft 734-744.

> "This algorithm employs two cascaded pre/postprocessors, one designed to recode a run of identical byte values, the other to detect and recode Unicode-like data sequences. Unicode [13] is used by the Java language and other Web-based applications."

Craft, 742.

> "The CRAM compression technology is clearly able to cover an

extremely wide range of applicability. Its small size allows integration into the smallest portable, handheld, or wireless applications, where it is much faster and consumes far less power than software."

Craft, 744.

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Craft, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Craft discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

See also

> "However, in our later designs, we chose to add a conventional address decoder to the CAM so that it can also function as an SRAM during an LZ1 decode function. This results in a very compact hardware encoder/decoder, which we call a CRAM design, as it combines a compression engine CAM and a decompression engine RAM into one silicon array."

Craft, 738.

> "An LZ1 decompressor builds and maintains an identical history copy, which it updates in the same manner as the encoder, as each LITERAL or COPY_POINTER is processed."

Craft, 738.

> "The code byte-stream output from this preprocessor is then fed into a standard ALDC-1 encoder. For decompression, an ALDC decoder generates code byte values which are then fed into a hardware postprocessor to reconstruct the original data bit stream."

Craft, 741.

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Craft, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Craft, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above.	

22.1 maintaining a list of boot data used for booting a computer system;.	Craft, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claim 1.1 above.	

22.2 initializing a central processing unit of the computer system;.	Craft, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claim 1.2 above.	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Craft, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claim 1.3 above.	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Craft, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claim 1.4 above.	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Craft, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 4, 10 and 11 above.	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Craft, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 15 and 22 above.	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Craft, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 16 and 22 above.	

27.1 a boot device..	Craft, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claim 1 above.	

27.2 a processor..	Craft, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claim 1.2 above.	

27.3 cache memory; and.	Craft, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.3 and 1.4 above.	

27.4 non-volatile memory for storing logic code for use by the processor,..	Craft, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claim 1.1 and 1.3 above.	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Craft, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claim 1.1 above.	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Craft, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claim 1.3 above.	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Craft, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claim 1.4 above.	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Craft, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 4, 10, and 11 above.	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Craft, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 15 and 27 above.	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Craft, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 16 and 27 above.	

Appendix A30
Invalidity of U.S. Patent 7,181,608 based on Douglis

Douglis, "One the Role of Compression in Distributed Systems" ("Douglis 1") and Douglis, "The Compression Cache: Using On-Line Compression to Extend Physical Memory," Winter 1993 USENIX Conference, Jan 1993 ("Douglis 2") (collectively, "Douglis"), alone or in combination, invalidate claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A30
Invalidity of U.S. Patent 7,181,608 based on Douglis

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Douglis, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this claim limitation:

Abstract

Compression has been used in numerous ways for many years, but recently two factors have combined in a way to push compression to the forefront of distributed systems. First, the disparity between processor speeds and I/O rates is ever-increasing, making it possible to perform compression in software to a much greater extent than was previously feasible. Second, the growth of new applications demanding enormous data rates, such as digital video and audio, makes hardware compression increasingly desirable. I discuss the importance of compression in various environments and describe how compression may be used not only to reduce the demand for disk space, disk bandwidth, and network bandwidth, but also to appear to extend physical memory.

Douglis 1, Abstact

Data compression has long been used as a method to reduce the number of bits being stored or transmitted [7], but stored where and transmitted over what? The answer to this question is changing as the gap between processing speed and I/O bandwidth increases, and the uses for computers shift toward multimedia and other high-data-rate applications.

Traditionally, compression was used to reduce disk storage demands, by selectively compressing files that had not been accessed for a long time, and to reduce bandwidth requirements over such media as wide-area networks and modems. Some forms of compression are now nearly ubiquitous: for example, the vast majority of "anonymous FTP" archives store all but the most trivial of files in compressed format, saving disk space while making retrievals over the Internet faster. (Other sites sometimes store the original files in uncompressed format but allow on-the-fly compression upon request.) Images impose especially

Douglis 1, at 1

Even today, compression buys a significant improvement in disk capacity with only a slight degradation in performance. Cate and Gross combined compression and caching in a two-level file store, which automatically compresses least-recently-used files in order to save disk space [3]. Stacker[1] uses compression to increase the effective capacity of a PC hard disk, sometimes improving I/O performance but often being measurably slower than without compression; the slowness is due to the relative performance of an Intel 386 chip compared to a typical PC hard drive, and can be improved with a special processor board [5]. Taunton described a mechanism for compressing binary executables on Acorn personal computers, reducing disk space requirements and improving file system bandwidth; since only decompression was being performed on-line, the processing overhead was offset by the reduction in I/O [10]. Burrows, et al., combined on-line compression with Sprite LFS [9] to compress and decompress all disk I/O automatically [2]. They used it primarily to increase effective disk capacity but expect that with hardware compression, the increase in effective disk bandwidth would improve overall system performance.

Douglis 1, 2

memory can allow a user to run larger processes without buying more memory. In this case, virtual memory pages would be compressed and written back into physical memory rather than to a backing store. This idea was briefly mentioned by Appel and Li, who claimed that user-level support for compression would be more effective than generic operating system support [1]. Nevertheless, I believe that OS support for compressed virtual memory can prove useful. I refer to the technique of trading regular physical memory for data stored in compressed format as a *compression cache* (CCACHE).

One question, of course, is whether paging is even interesting or desirable. In some environments, paging is largely passé due to the large amounts of memory available [4]. However, I believe there are still systems and applications that could benefit greatly from a cheaper method of paging. Mobile computers are especially likely to have less memory available than needed, and the cost of paging over a wireless network or onto a small local disk might be much more than the cost of compressing the page. Even in a workstation-based environment, a main-memory database that is within a factor of two or three of the size of available physical memory might fit completely in memory in compressed format. The important issues are how effective compression is at saving memory and how costly compression is by comparison to network or disk I/O.

In the best case, compression can be done in hardware, and the cost of compression will be limited only by memory-to-memory bandwidth. In this case I would expect the CCACHE to provide a substantial improvement in performance. Beyond this, however, I argue that on-line compression in software can still provide benefits to a wide class of applications if designed properly. In this section I describe an implementation of the CCACHE in software, and the performance of some benchmarks, to support this claim.

Douglis 1, 2

There are a number of potential benefits of using a compression cache. First, compressing a page and retaining it in memory can be cheaper than writing it to backing store. Second, reading it back from memory is likely to be much cheaper than reading it from backing store, even taking the cost of decompression into account. Finally, if the page is eventually written to backing store, it can be written in compressed format, requiring less bandwidth.

The CCACHE has some potential disadvantages as well. Most importantly, it competes with user processes (and perhaps the file system cache) for memory. Putting a page in the CCACHE has the effect of freeing up only part of a page, so more pages must be transferred from uncompressed memory to the CCACHE in order to make room for one new page. Processes will take additional page faults because they are given less memory. (This suggests that the CCACHE should be of a variable size, and should be used only when its presence improves performance rather than degrading it.) Additionally, the CCACHE adds overhead in the form of added complexity during page faults, as well as extra code and data associated with the cache.

Douglis 1, 3.1

Douglis **Claim 1 (Preamble)**

"A method for providing accelerated loading of an operating system, comprising the steps of:"

As was mentioned above, the CCACHE is primarily targeted for mobile computers, which typically have less memory than desk-based computers of the same generation. Nevertheless, the idea of the CCACHE extends naturally to any environment that pages when applications use more virtual memory than physical memory. I have added a compression cache to the Sprite operating system [8], running on DECstation[2] 5000/200 workstations (approximately 18 SPECmarks, running a 25 MHz MIPS R3000 processor). Sprite is a particularly suitable system for the CCACHE, because it already supports the idea of trading physical memory between the virtual memory system and the file system [6]. Trading memory a third way, including the compression cache, would be a natural extension of this technique.

The initial prototype of the compression cache has been simplified in a number of ways. First of all, I use a fixed-size CCACHE, which is configured at kernel load time. The fixed-size cache is simpler but results in unneeded overhead for applications that would otherwise fit in memory; measurements of the effects of this are presented below. Secondly, compressed pages are written out using the same number of bytes as uncompressed pages: a full 4-Kbyte file block. In the future compressed pages will be combined into a smaller number of file blocks to save disk bandwidth. Third, I use only a single LZ-based compression algorithm [11] for all data. The compression algorithm reduces pages to 30–40% of their original size on average, depending on the application mix; I would expect application-specific compression algorithms to do much better. Finally, in order to estimate the effects of compression in a system with a large gap between processor and I/O speed, I configured the test system to page to a local disk running the original Sprite file system [6] rather than the faster Sprite LFS [9].

Douglis 1, 3.2

As a second test, I ran an application that responds to queries based on the contents of a hash table. The database containing the hash table was over 40 Mbytes, all of which was cached in memory, and the total address space of the process was over 60 Mbytes. The benchmark consisted of running a set of 9 queries against this database; the queries were run once to load the process's address space with the appropriate parts of its database and a second time to evaluate performance. The benchmark was run on the same DECstation as the previous measurements, but without the artificial restriction, so it had about 25 Mbytes of memory available. Running with a 12-Mbyte CCACHE was 32% faster than without the compression cache (9.28 minutes versus 13.7).

Douglis 1, 3.4

Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2, Abstract

4 Design

This section describes the design and implementation of a compression cache in Sprite. Sprite is largely compatible with 4.3 BSD UNIX, but its virtual memory system has an interesting difference from most versions of UNIX: physical memory is traded dynamically between VM for application processes and the file system's buffer cache [9]. Since the compression cache must vary in size dynamically as well, Sprite provides a good framework for prototyping the compression cache. The idea of the compression cache should extend naturally to UNIX,[3] Mach, or other systems; in fact, Mach's external pager interface [7] should be an excellent foundation for future work in this area.

The target environment for this research consists of mobile computers with limited memory and network bandwidth, and with small local disks or no disks at all. Because of the limited availability of Sprite, the compression cache has been prototyped in a workstation environment, running on DECstation 5000/200 workstations, paging to a local RZ57 disk. The Sprite kernel is configurable at boot-time to allow the system to use a variable amount of physical memory, so a 32-Mbyte machine can behave as though it has as little as 12 Mbytes. About 6 Mbytes are used by the kernel for code, page tables, and some forms of tracing that cannot currently be disabled.

Douglis 2, 4

1.1 maintaining a list of boot data used for booting a computer system;	Douglis, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this claim limitation:

Abstract

Compression has been used in numerous ways for many years, but recently two factors have combined in a way to push compression to the forefront of distributed systems. First, the disparity between processor speeds and I/O rates is ever-increasing, making it possible to perform compression in software to a much greater extent than was previously feasible. Second, the growth of new applications demanding enormous data rates, such as digital video and audio, makes hardware compression increasingly desirable. I discuss the importance of compression in various environments and describe how compression may be used not only to reduce the demand for disk space, disk bandwidth, and network bandwidth, but also to appear to extend physical memory.

Douglis 1, Abstact

As was mentioned above, the CCACHE is primarily targeted for mobile computers, which typically have less memory than desk-based computers of the same generation. Nevertheless, the idea of the CCACHE extends naturally to any environment that pages when applications use more virtual memory than physical memory. I have added a compression cache to the Sprite operating system [8], running on DECstation[2] 5000/200 workstations (approximately 18 SPECmarks, running a 25 MHz MIPS R3000 processor). Sprite is a particularly suitable system for the CCACHE, because it already supports the idea of trading physical memory between the virtual memory system and the file system [6]. Trading memory a third way, including the compression cache, would be a natural extension of this technique.

The initial prototype of the compression cache has been simplified in a number of ways. First of all, I use a fixed-size CCACHE, which is configured at kernel load time. The fixed-size cache is simpler but results in unneeded overhead for applications that would otherwise fit in memory; measurements of the effects of this are presented below. Secondly, compressed pages are written out using the same number of bytes as uncompressed pages: a full 4-Kbyte file block. In the future compressed pages will be combined into a smaller number of file blocks to save disk bandwidth. Third, I use only a single LZ-based compression algorithm [11] for all data. The compression algorithm reduces pages to 30–40% of their original size on average, depending on the application mix; I would expect application-specific compression algorithms to do much better. Finally, in order to estimate the effects of compression in a system with a large gap between processor and I/O speed, I configured the test system to page to a local disk running the original Sprite file system [6] rather than the faster Sprite LFS [9].

Douglis 1, 3.2

Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2, Abstract

4 Design

This section describes the design and implementation of a compression cache in Sprite. Sprite is largely compatible with 4.3 BSD UNIX, but its virtual memory system has an interesting difference from most versions of UNIX: physical memory is traded dynamically between VM for application processes and the file system's buffer cache [9]. Since the compression cache must vary in size dynamically as well, Sprite provides a good framework for prototyping the compression cache. The idea of the compression cache should extend naturally to UNIX,[3] Mach, or other systems; in fact, Mach's external pager interface [7] should be an excellent foundation for future work in this area.

The target environment for this research consists of mobile computers with limited memory and network bandwidth, and with small local disks or no disks at all. Because of the limited availability of Sprite, the compression cache has been prototyped in a workstation environment, running on DECstation 5000/200 workstations, paging to a local RZ57 disk. The Sprite kernel is configurable at boot-time to allow the system to use a variable amount of physical memory, so a 32-Mbyte machine can behave as though it has as little as 12 Mbytes. About 6 Mbytes are used by the kernel for code, page tables, and some forms of tracing that cannot currently be disabled.

Douglis 2, 4

1.2 initializing a central processing unit of the computer system;	Douglis, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this claim limitation:

Abstract

Compression has been used in numerous ways for many years, but recently two factors have combined in a way to push compression to the forefront of distributed systems. First, the disparity between processor speeds and I/O rates is ever-increasing, making it possible to perform compression in software to a much greater extent than was previously feasible. Second, the growth of new applications demanding enormous data rates, such as digital video and audio, makes hardware compression increasingly desirable. I discuss the importance of compression in various environments and describe how compression may be used not only to reduce the demand for disk space, disk bandwidth, and network bandwidth, but also to appear to extend physical memory.

Douglis 1, Abstact

Data compression has long been used as a method to reduce the number of bits being stored or transmitted [7], but stored where and transmitted over what? The answer to this question is changing as the gap between processing speed and I/O bandwidth increases, and the uses for computers shift toward multimedia and other high-data-rate applications.

Traditionally, compression was used to reduce disk storage demands, by selectively compressing files that had not been accessed for a long time, and to reduce bandwidth requirements over such media as wide-area networks and modems. Some forms of compression are now nearly ubiquitous: for example, the vast majority of "anonymous FTP" archives store all but the most trivial of files in compressed format, saving disk space while making retrievals over the Internet faster. (Other sites sometimes store the original files in uncompressed format but allow on-the-fly compression upon request.) Images impose especially

Douglis 1, at 1

Even today, compression buys a significant improvement in disk capacity with only a slight degradation in performance. Cate and Gross combined compression and caching in a two-level file store, which automatically compresses least-recently-used files in order to save disk space [3]. Stacker[1] uses compression to increase the effective capacity of a PC hard disk, sometimes improving I/O performance but often being measurably slower than without compression; the slowness is due to the relative performance of an Intel 386 chip compared to a typical PC hard drive, and can be improved with a special processor board [5]. Taunton described a mechanism for compressing binary executables on Acorn personal computers, reducing disk space requirements and improving file system bandwidth; since only decompression was being performed on-line, the processing overhead was offset by the reduction in I/O [10]. Burrows, et al., combined on-line compression with Sprite LFS [9] to compress and decompress all disk I/O automatically [2]. They used it primarily to increase effective disk capacity but expect that with hardware compression, the increase in effective disk bandwidth would improve overall system performance.

Douglis 1, 2

memory can allow a user to run larger processes without buying more memory. In this case, virtual memory pages would be compressed and written back into physical memory rather than to a backing store. This idea was briefly mentioned by Appel and Li, who claimed that user-level support for compression would be more effective than generic operating system support [1]. Nevertheless, I believe that OS support for compressed virtual memory can prove useful. I refer to the technique of trading regular physical memory for data stored in compressed format as a *compression cache* (CCACHE).

One question, of course, is whether paging is even interesting or desirable. In some environments, paging is largely passé due to the large amounts of memory available [4]. However, I believe there are still systems and applications that could benefit greatly from a cheaper method of paging. Mobile computers are especially likely to have less memory available than needed, and the cost of paging over a wireless network or onto a small local disk might be much more than the cost of compressing the page. Even in a workstation-based environment, a main-memory database that is within a factor of two or three of the size of available physical memory might fit completely in memory in compressed format. The important issues are how effective compression is at saving memory and how costly compression is by comparison to network or disk I/O.

In the best case, compression can be done in hardware, and the cost of compression will be limited only by memory-to-memory bandwidth. In this case I would expect the CCACHE to provide a substantial improvement in performance. Beyond this, however, I argue that on-line compression in software can still provide benefits to a wide class of applications if designed properly. In this section I describe an implementation of the CCACHE in software, and the performance of some benchmarks, to support this claim.

Douglis 1, 2

There are a number of potential benefits of using a compression cache. First, compressing a page and retaining it in memory can be cheaper than writing it to backing store. Second, reading it back from memory is likely to be much cheaper than reading it from backing store, even taking the cost of decompression into account. Finally, if the page is eventually written to backing store, it can be written in compressed format, requiring less bandwidth.

The CCACHE has some potential disadvantages as well. Most importantly, it competes with user processes (and perhaps the file system cache) for memory. Putting a page in the CCACHE has the effect of freeing up only part of a page, so more pages must be transferred from uncompressed memory to the CCACHE in order to make room for one new page. Processes will take additional page faults because they are given less memory. (This suggests that the CCACHE should be of a variable size, and should be used only when its presence improves performance rather than degrading it.) Additionally, the CCACHE adds overhead in the form of added complexity during page faults, as well as extra code and data associated with the cache.

Douglis 1, 3.1

As was mentioned above, the CCACHE is primarily targeted for mobile computers, which typically have less memory than desk-based computers of the same generation. Nevertheless, the idea of the CCACHE extends naturally to any environment that pages when applications use more virtual memory than physical memory. I have added a compression cache to the Sprite operating system [8], running on DECstation[2] 5000/200 workstations (approximately 18 SPECmarks, running a 25 MHz MIPS R3000 processor). Sprite is a particularly suitable system for the CCACHE, because it already supports the idea of trading physical memory between the virtual memory system and the file system [6]. Trading memory a third way, including the compression cache, would be a natural extension of this technique.

The initial prototype of the compression cache has been simplified in a number of ways. First of all, I use a fixed-size CCACHE, which is configured at kernel load time. The fixed-size cache is simpler but results in unneeded overhead for applications that would otherwise fit in memory; measurements of the effects of this are presented below. Secondly, compressed pages are written out using the same number of bytes as uncompressed pages: a full 4-Kbyte file block. In the future compressed pages will be combined into a smaller number of file blocks to save disk bandwidth. Third, I use only a single LZ-based compression algorithm [11] for all data. The compression algorithm reduces pages to 30–40% of their original size on average, depending on the application mix; I would expect application-specific compression algorithms to do much better. Finally, in order to estimate the effects of compression in a system with a large gap between processor and I/O speed, I configured the test system to page to a local disk running the original Sprite file system [6] rather than the faster Sprite LFS [9].

Douglis 1, 3.2

As a second test, I ran an application that responds to queries based on the contents of a hash table. The database containing the hash table was over 40 Mbytes, all of which was cached in memory, and the total address space of the process was over 60 Mbytes. The benchmark consisted of running a set of 9 queries against this database; the queries were run once to load the process's address space with the appropriate parts of its database and a second time to evaluate performance. The benchmark was run on the same DECstation as the previous measurements, but without the artificial restriction, so it had about 25 Mbytes of memory available. Running with a 12-Mbyte CCACHE was 32% faster than without the compression cache (9.28 minutes versus 13.7).

Douglis 1, 3.4

Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2, Abstract

Abstract

Compression has been used in numerous ways for many years, but recently two factors have combined in a way to push compression to the forefront of distributed systems. First, the disparity between processor speeds and I/O rates is ever-increasing, making it possible to perform compression in software to a much greater extent than was previously feasible. Second, the growth of new applications demanding enormous data rates, such as digital video and audio, makes hardware compression increasingly desirable. I discuss the importance of compression in various environments and describe how compression may be used not only to reduce the demand for disk space, disk bandwidth, and network bandwidth, but also to appear to extend physical memory.

Douglis 1, Abstact

As was mentioned above, the CCACHE is primarily targeted for mobile computers, which typically have less memory than desk-based computers of the same generation. Nevertheless, the idea of the CCACHE extends naturally to any environment that pages when applications use more virtual memory than physical memory. I have added a compression cache to the

Sprite operating system [8], running on DECstation[2] 5000/200 workstations (approximately 18 SPECmarks, running a 25 MHz MIPS R3000 processor). Sprite is a particularly suitable system for the CCACHE, because it already supports the idea of trading physical memory between the virtual memory system and the file system [6]. Trading memory a third way, including the compression cache, would be a natural extension of this technique.

The initial prototype of the compression cache has been simplified in a number of ways. First of all, I use a fixed-size CCACHE, which is configured at kernel load time. The fixed-size cache is simpler but results in unneeded overhead for applications that would otherwise fit in memory; measurements of the effects of this are presented below. Secondly, compressed pages are written out using the same number of bytes as uncompressed pages: a full 4-Kbyte file block. In the future compressed pages will be combined into a smaller number of file blocks to save disk bandwidth. Third, I use only a single LZ-based compression algorithm [11] for all data. The compression algorithm reduces pages to 30–40% of their original size on average, depending on the application mix; I would expect application-specific compression algorithms to do much better. Finally, in order to estimate the effects of compression in a system with a large gap between processor and I/O speed, I configured the test system to page to a local disk running the original Sprite file system [6] rather than the faster Sprite LFS [9].

Douglis 1, 3.2

Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2, Abstract

4 Design

This section describes the design and implementation of a compression cache in Sprite. Sprite is largely compatible with 4.3 BSD UNIX, but its virtual memory system has an interesting difference from most versions of UNIX: physical memory is traded dynamically between VM for application processes and the file system's buffer cache [9]. Since the compression cache must vary in size dynamically as well, Sprite provides a good framework for prototyping the compression cache. The idea of the compression cache should extend naturally to UNIX,[3] Mach, or other systems; in fact, Mach's external pager interface [7] should be an excellent foundation for future work in this area.

The target environment for this research consists of mobile computers with limited memory and network bandwidth, and with small local disks or no disks at all. Because of the limited availability of Sprite, the compression cache has been prototyped in a workstation environment, running on DECstation 5000/200 workstations, paging to a local RZ57 disk. The Sprite kernel is configurable at boot-time to allow the system to use a variable amount of physical memory, so a 32-Mbyte machine can behave as though it has as little as 12 Mbytes. About 6 Mbytes are used by the kernel for code, page tables, and some forms of tracing that cannot currently be disabled.

Douglis 2, 4

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Douglis, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this claim limitation:

Abstract

Compression has been used in numerous ways for many years, but recently two factors have combined in a way to push compression to the forefront of distributed systems. First, the disparity between processor speeds and I/O rates is ever-increasing, making it possible to perform compression in software to a much greater extent than was previously feasible. Second, the growth of new applications demanding enormous data rates, such as digital video and audio, makes hardware compression increasingly desirable. I discuss the importance of compression in various environments and describe how compression may be used not only to reduce the demand for disk space, disk bandwidth, and network bandwidth, but also to appear to extend physical memory.

Douglis 1, Abstact

Data compression has long been used as a method to reduce the number of bits being stored or transmitted [7], but stored where and transmitted over what? The answer to this question is changing as the gap between processing speed and I/O bandwidth increases, and the uses for computers shift toward multimedia and other high-data-rate applications.

Traditionally, compression was used to reduce disk storage demands, by selectively compressing files that had not been accessed for a long time, and to reduce bandwidth requirements over such media as wide-area networks and modems. Some forms of compression are now nearly ubiquitous: for example, the vast majority of "anonymous FTP" archives store all but the most trivial of files in compressed format, saving disk space while making retrievals over the Internet faster. (Other sites sometimes store the original files in uncompressed format but allow on-the-fly compression upon request.) Images impose especially

Douglis 1, at 1

Even today, compression buys a significant improvement in disk capacity with only a slight degradation in performance. Cate and Gross combined compression and caching in a two-level file store, which automatically compresses least-recently-used files in order to save disk space [3]. Stacker[1] uses compression to increase the effective capacity of a PC hard disk, sometimes improving I/O performance but often being measurably slower than without compression; the slowness is due to the relative performance of an Intel 386 chip compared to a typical PC hard drive, and can be improved with a special processor board [5]. Taunton described a mechanism for compressing binary executables on Acorn personal computers, reducing disk space requirements and improving file system bandwidth; since only decompression was being performed on-line, the processing overhead was offset by the reduction in I/O [10]. Burrows, et al., combined on-line compression with Sprite LFS [9] to compress and decompress all disk I/O automatically [2]. They used it primarily to increase effective disk capacity but expect that with hardware compression, the increase in effective disk bandwidth would improve overall system performance.

Douglis 1, 2

memory can allow a user to run larger processes without buying more memory. In this case, virtual memory pages would be compressed and written back into physical memory rather than to a backing store. This idea was briefly mentioned by Appel and Li, who claimed that user-level support for compression would be more effective than generic operating system support [1]. Nevertheless, I believe that OS support for compressed virtual memory can prove useful. I refer to the technique of trading regular physical memory for data stored in compressed format as a *compression cache* (CCACHE).

One question, of course, is whether paging is even interesting or desirable. In some environments, paging is largely passé due to the large amounts of memory available [4]. However, I believe there are still systems and applications that could benefit greatly from a cheaper method of paging. Mobile computers are especially likely to have less memory available than needed, and the cost of paging over a wireless network or onto a small local disk might be much more than the cost of compressing the page. Even in a workstation-based environment, a main-memory database that is within a factor of two or three of the size of available physical memory might fit completely in memory in compressed format. The important issues are how effective compression is at saving memory and how costly compression is by comparison to network or disk I/O.

In the best case, compression can be done in hardware, and the cost of compression will be limited only by memory-to-memory bandwidth. In this case I would expect the CCACHE to provide a substantial improvement in performance. Beyond this, however, I argue that on-line compression in software can still provide benefits to a wide class of applications if designed properly. In this section I describe an implementation of the CCACHE in software, and the performance of some benchmarks, to support this claim.

Douglis 1, 2

There are a number of potential benefits of using a compression cache. First, compressing a page and retaining it in memory can be cheaper than writing it to backing store. Second, reading it back from memory is likely to be much cheaper than reading it from backing store, even taking the cost of decompression into account. Finally, if the page is eventually written to backing store, it can be written in compressed format, requiring less bandwidth.

The CCACHE has some potential disadvantages as well. Most importantly, it competes with user processes (and perhaps the file system cache) for memory. Putting a page in the CCACHE has the effect of freeing up only part of a page, so more pages must be transferred from uncompressed memory to the CCACHE in order to make room for one new page. Processes will take additional page faults because they are given less memory. (This suggests that the CCACHE should be of a variable size, and should be used only when its presence improves performance rather than degrading it.) Additionally, the CCACHE adds overhead in the form of added complexity during page faults, as well as extra code and data associated with the cache.

Douglis 1, 3.1

Douglis Claim 1.3

"preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

As was mentioned above, the CCACHE is primarily targeted for mobile computers, which typically have less memory than desk-based computers of the same generation. Nevertheless, the idea of the CCACHE extends naturally to any environment that pages when applications use more virtual memory than physical memory. I have added a compression cache to the Sprite operating system [8], running on DECstation[2] 5000/200 workstations (approximately 18 SPECmarks, running a 25 MHz MIPS R3000 processor). Sprite is a particularly suitable system for the CCACHE, because it already supports the idea of trading physical memory between the virtual memory system and the file system [6]. Trading memory a third way, including the compression cache, would be a natural extension of this technique.

The initial prototype of the compression cache has been simplified in a number of ways. First of all, I use a fixed-size CCACHE, which is configured at kernel load time. The fixed-size cache is simpler but results in unneeded overhead for applications that would otherwise fit in memory; measurements of the effects of this are presented below. Secondly, compressed pages are written out using the same number of bytes as uncompressed pages: a full 4-Kbyte file block. In the future compressed pages will be combined into a smaller number of file blocks to save disk bandwidth. Third, I use only a single LZ-based compression algorithm [11] for all data. The compression algorithm reduces pages to 30–40% of their original size on average, depending on the application mix; I would expect application-specific compression algorithms to do much better. Finally, in order to estimate the effects of compression in a system with a large gap between processor and I/O speed, I configured the test system to page to a local disk running the original Sprite file system [6] rather than the faster Sprite LFS [9].

Douglis 1, 3.2

As a second test, I ran an application that responds to queries based on the contents of a hash table. The database containing the hash table was over 40 Mbytes, all of which was cached in memory, and the total address space of the process was over 60 Mbytes. The benchmark consisted of running a set of 9 queries against this database; the queries were run once to load the process's address space with the appropriate parts of its database and a second time to evaluate performance. The benchmark was run on the same DECstation as the previous measurements, but without the artificial restriction, so it had about 25 Mbytes of memory available. Running with a 12-Mbyte CCACHE was 32% faster than without the compression cache (9.28 minutes versus 13.7).

Douglis 1, 3.4

Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on-line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2, Abstract

4 Design

This section describes the design and implementation of a compression cache in Sprite. Sprite is largely compatible with 4.3 BSD UNIX, but its virtual memory system has an interesting difference from most versions of UNIX: physical memory is traded dynamically between VM for application processes and the file system's buffer cache [9]. Since the compression cache must vary in size dynamically as well, Sprite provides a good framework for prototyping the compression cache. The idea of the compression cache should extend naturally to UNIX,[3] Mach, or other systems; in fact, Mach's external pager interface [7] should be an excellent foundation for future work in this area.

The target environment for this research consists of mobile computers with limited memory and network bandwidth, and with small local disks or no disks at all. Because of the limited availability of Sprite, the compression cache has been prototyped in a workstation environment, running on DECstation 5000/200 workstations, paging to a local RZ57 disk. The Sprite kernel is configurable at boot-time to allow the system to use a variable amount of physical memory, so a 32-Mbyte machine can behave as though it has as little as 12 Mbytes. About 6 Mbytes are used by the kernel for code, page tables, and some forms of tracing that cannot currently be disabled.

Douglis 2, 4

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Douglis, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this claim limitation:

Abstract

Compression has been used in numerous ways for many years, but recently two factors have combined in a way to push compression to the forefront of distributed systems. First, the disparity between processor speeds and I/O rates is ever-increasing, making it possible to perform compression in software to a much greater extent than was previously feasible. Second, the growth of new applications demanding enormous data rates, such as digital video and audio, makes hardware compression increasingly desirable. I discuss the importance of compression in various environments and describe how compression may be used not only to reduce the demand for disk space, disk bandwidth, and network bandwidth, but also to appear to extend physical memory.

Douglis 1, Abstact

Data compression has long been used as a method to reduce the number of bits being stored or transmitted [7], but stored where and transmitted over what? The answer to this question is changing as the gap between processing speed and I/O bandwidth increases, and the uses for computers shift toward multimedia and other high-data-rate applications.

Traditionally, compression was used to reduce disk storage demands, by selectively compressing files that had not been accessed for a long time, and to reduce bandwidth requirements over such media as wide-area networks and modems. Some forms of compression are now nearly ubiquitous: for example, the vast majority of "anonymous FTP" archives store all but the most trivial of files in compressed format, saving disk space while making retrievals over the Internet faster. (Other sites sometimes store the original files in uncompressed format but allow on-the-fly compression upon request.) Images impose especially

Douglis 1, at 1

Even today, compression buys a significant improvement in disk capacity with only a slight degradation in performance. Cate and Gross combined compression and caching in a two-level file store, which automatically compresses least-recently-used files in order to save disk space [3]. Stacker[1] uses compression to increase the effective capacity of a PC hard disk, sometimes improving I/O performance but often being measurably slower than without compression; the slowness is due to the relative performance of an Intel 386 chip compared to a typical PC hard drive, and can be improved with a special processor board [5]. Taunton described a mechanism for compressing binary executables on Acorn personal computers, reducing disk space requirements and improving file system bandwidth; since only decompression was being performed on-line, the processing overhead was offset by the reduction in I/O [10]. Burrows, et al., combined on-line compression with Sprite LFS [9] to compress and decompress all disk I/O automatically [2]. They used it primarily to increase effective disk capacity but expect that with hardware compression, the increase in effective disk bandwidth would improve overall system performance.

Douglis 1, 2

memory can allow a user to run larger processes without buying more memory. In this case, virtual memory pages would be compressed and written back into physical memory rather than to a backing store. This idea was briefly mentioned by Appel and Li, who claimed that user-level support for compression would be more effective than generic operating system support [1]. Nevertheless, I believe that OS support for compressed virtual memory can prove useful. I refer to the technique of trading regular physical memory for data stored in compressed format as a *compression cache* (CCACHE).

One question, of course, is whether paging is even interesting or desirable. In some environments, paging is largely passé due to the large amounts of memory available [4]. However, I believe there are still systems and applications that could benefit greatly from a cheaper method of paging. Mobile computers are especially likely to have less memory available than needed, and the cost of paging over a wireless network or onto a small local disk might be much more than the cost of compressing the page. Even in a workstation-based environment, a main-memory database that is within a factor of two or three of the size of available physical memory might fit completely in memory in compressed format. The important issues are how effective compression is at saving memory and how costly compression is by comparison to network or disk I/O.

In the best case, compression can be done in hardware, and the cost of compression will be limited only by memory-to-memory bandwidth. In this case I would expect the CCACHE to provide a substantial improvement in performance. Beyond this, however, I argue that on-line compression in software can still provide benefits to a wide class of applications if designed properly. In this section I describe an implementation of the CCACHE in software, and the performance of some benchmarks, to support this claim.

Douglis 1, 2

Douglis Claim 1.4

"servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

There are a number of potential benefits of using a compression cache. First, compressing a page and retaining it in memory can be cheaper than writing it to backing store. Second, reading it back from memory is likely to be much cheaper than reading it from backing store, even taking the cost of decompression into account. Finally, if the page is eventually written to backing store, it can be written in compressed format, requiring less bandwidth.

The CCACHE has some potential disadvantages as well. Most importantly, it competes with user processes (and perhaps the file system cache) for memory. Putting a page in the CCACHE has the effect of freeing up only part of a page, so more pages must be transferred from uncompressed memory to the CCACHE in order to make room for one new page. Processes will take additional page faults because they are given less memory. (This suggests that the CCACHE should be of a variable size, and should be used only when its presence improves performance rather than degrading it.) Additionally, the CCACHE adds overhead in the form of added complexity during page faults, as well as extra code and data associated with the cache.

Douglis 1, 3.1

As was mentioned above, the CCACHE is primarily targeted for mobile computers, which typically have less memory than desk-based computers of the same generation. Nevertheless, the idea of the CCACHE extends naturally to any environment that pages when applications use more virtual memory than physical memory. I have added a compression cache to the Sprite operating system [8], running on DECstation[2] 5000/200 workstations (approximately 18 SPECmarks, running a 25 MHz MIPS R3000 processor). Sprite is a particularly suitable system for the CCACHE, because it already supports the idea of trading physical memory between the virtual memory system and the file system [6]. Trading memory a third way, including the compression cache, would be a natural extension of this technique.

The initial prototype of the compression cache has been simplified in a number of ways. First of all, I use a fixed-size CCACHE, which is configured at kernel load time. The fixed-size cache is simpler but results in unneeded overhead for applications that would otherwise fit in memory; measurements of the effects of this are presented below. Secondly, compressed pages are written out using the same number of bytes as uncompressed pages: a full 4-Kbyte file block. In the future compressed pages will be combined into a smaller number of file blocks to save disk bandwidth. Third, I use only a single LZ-based compression algorithm [11] for all data. The compression algorithm reduces pages to 30–40% of their original size on average, depending on the application mix; I would expect application-specific compression algorithms to do much better. Finally, in order to estimate the effects of compression in a system with a large gap between processor and I/O speed, I configured the test system to page to a local disk running the original Sprite file system [6] rather than the faster Sprite LFS [9].

Douglis 1, 3.2

As a second test, I ran an application that responds to queries based on the contents of a hash table. The database containing the hash table was over 40 Mbytes, all of which was cached in memory, and the total address space of the process was over 60 Mbytes. The benchmark consisted of running a set of 9 queries against this database; the queries were run once to load the process's address space with the appropriate parts of its database and a second time to evaluate performance. The benchmark was run on the same DECstation as the previous measurements, but without the artificial restriction, so it had about 25 Mbytes of memory available. Running with a 12-Mbyte CCACHE was 32% faster than without the compression cache (9.28 minutes versus 13.7).

Douglis 1, 3.4

Appendix A30
Invalidity of U.S. Patent 7,181,608 based on Douglis

Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2, Abstract

4 Design

This section describes the design and implementation of a compression cache in Sprite. Sprite is largely compatible with 4.3 BSD UNIX, but its virtual memory system has an interesting difference from most versions of UNIX: physical memory is traded dynamically between VM for application processes and the file system's buffer cache [9]. Since the compression cache must vary in size dynamically as well, Sprite provides a good framework for prototyping the compression cache. The idea of the compression cache should extend naturally to UNIX,[3] Mach, or other systems; in fact, Mach's external pager interface [7] should be an excellent foundation for future work in this area.

The target environment for this research consists of mobile computers with limited memory and network bandwidth, and with small local disks or no disks at all. Because of the limited availability of Sprite, the compression cache has been prototyped in a workstation environment, running on DECstation 5000/200 workstations, paging to a local RZ57 disk. The Sprite kernel is configurable at boot-time to allow the system to use a variable amount of physical memory, so a 32-Mbyte machine can behave as though it has as little as 12 Mbytes. About 6 Mbytes are used by the kernel for code, page tables, and some forms of tracing that cannot currently be disabled.

Douglis 2, 4

Douglis Claim 1.4

"servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Douglis, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

3.3 Target Applications

I next address the issue of what applications might make effective use of a compression cache. Obviously, many applications will fit into memory without the need for compression, and other applications may cause excessive I/O even with compression. The applications that can best make use of a CCACHE are those that will not comfortably fit into physical memory without compression but will fit if some of their pages are compressed.

One can imagine a contrived scenario in which the CCACHE would provide significant performance benefits: if an application cycles linearly through a working set that is one page larger than the maximum number of pages allowed to be resident, and a least-recently-used algorithm is used for page replacement, then the process will take a page fault on each new page. With the compression cache, the process will still fault on each page, but each fault will be satisfied by a compression and a decompression rather than a pair of disk I/Os.

Another scenario, as mentioned above, is the application that spreads its accesses uniformly in an address space much larger than physical memory. Although taking some of memory for the CCACHE will result in additional page faults that would otherwise not occur, the average cost of a page fault will be less with the compression cache in use as long as compression is much less expensive than disk I/O (say, one-third the cost) and the hit rate in the compression cache is reasonably high (say, 80–90%).

Douglis 1

Appendix A30
Invalidity of U.S. Patent 7,181,608 based on Douglis

Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on-line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2

1 Introduction

Over the past decade, the processing power and physical memory size of typical computers have increased dramatically. Even as workstation memory sizes are increasing, however, a new technology trend is pushing toward small memories: mobile computers that are smaller than their desk-top counterparts and are typically configured with significantly less memory. Application designers are sometimes forced to squeeze their applications to fit into available memory, and may not succeed. Therefore, in a general-purpose mobile computer, as with many computers, paging is needed to enable a wider range of applications to run—as long as it can be performed efficiently.

Douglis 2

The potential benefits of the compression cache depend on the relationship between the speed of compression and the I/O bandwidth of the system, as well as the compression ratio (anywhere from barely over 1:1 to about 4:1 in the experiments reported below). If the cost of compressing and copying a page were negligible, and pages compressed well, the compression cache could be used to give a computer the appearance of having additional physical memory. In practice, compressing and copying have costs associated with them, and the benefit of reducing traffic to the backing store is offset by the overhead of the compression cache. Overhead comes not only from the compression itself but from the additional page faults an application will experience when some memory is used for compressed pages (as well as the data structures used to support compressed pages). Furthermore, as mentioned above, not all applications compress well: for poorly suited applications, the effort to compress memory will be wasted and degrade rather than improve performance. Thus, depending on the application and the hardware environment, the benefits of reduced I/O may outweigh the costs of compression and additional faults, or vice-versa. Configuring the compression cache to improve performance in the first case while staying out of the way in the second case is an interesting, and difficult, problem.

Douglis 2

Douglis

Claim 2

"The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Douglis, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

The method for choosing when to grow or shrink the compression cache is similar to the algorithm in Sprite for trading memory between the file system and VM system. Sprite compares the age of the least-recently-used file block to the age of the LRU VM page, and reclaims the older of the two, modulo an adjustment to favor retaining VM pages longer. This "penalty" to the file system helps improve interactive performance, by preventing a large file from flushing a process's address space completely out of memory [10].

Douglis 2

Traditionally, compression was used to reduce disk storage demands, by selectively compressing files that had not been accessed for a long time, and to reduce bandwidth requirements over such media as wide-area networks and modems. Some forms of compression are now nearly ubiquitous: for example, the vast majority of "anonymous FTP" archives store all but the most trivial of files in compressed format, saving disk space while making retrievals over the Internet faster. (Other sites sometimes store the original files in uncompressed format but allow on-the-fly compression upon request.) Images impose especially excessive requirements on disk storage, disk bandwidth, and network bandwidth if they are not compressed; the current trend toward multimedia environments ensures that compression will play an increasingly important role in most distributed systems, even over only local-area networks.

Douglis 1

4. The method of claim 1, further comprising updating the list of boot data.	Douglis, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claim 1.1 above.	

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Douglis, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Douglis, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Douglis, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Douglis, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

The method for choosing when to grow or shrink the compression cache is similar to the algorithm in Sprite for trading memory between the file system and VM system. Sprite compares the age of the least-recently-used file block to the age of the LRU VM page, and reclaims the older of the two, modulo an adjustment to favor retaining VM pages longer. This "penalty" to the file system helps improve interactive performance, by preventing a large file from flushing a process's address space completely out of memory [10].

Douglis 2

Traditionally, compression was used to reduce disk storage demands, by selectively compressing files that had not been accessed for a long time, and to reduce bandwidth requirements over such media as wide-area networks and modems. Some forms of compression are now nearly ubiquitous: for example, the vast majority of "anonymous FTP" archives store all but the most trivial of files in compressed format, saving disk space while making retrievals over the Internet faster. (Other sites sometimes store the original files in uncompressed format but allow on-the-fly compression upon request.) Images impose especially excessive requirements on disk storage, disk bandwidth, and network bandwidth if they are not compressed; the current trend toward multimedia environments ensures that compression will play an increasingly important role in most distributed systems, even over only local-area networks.

Douglis 1

7.2 a cache memory device; and;	Douglis, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Douglis, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Douglis, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Douglis, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Douglis, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

3.3 Target Applications

I next address the issue of what applications might make effective use of a compression cache. Obviously, many applications will fit into memory without the need for compression, and other applications may cause excessive I/O even with compression. The applications that can best make use of a CCACHE are those that will not comfortably fit into physical memory without compression but will fit if some of their pages are compressed.

One can imagine a contrived scenario in which the CCACHE would provide significant performance benefits: if an application cycles linearly through a working set that is one page larger than the maximum number of pages allowed to be resident, and a least-recently-used algorithm is used for page replacement, then the process will take a page fault on each new page. With the compression cache, the process will still fault on each page, but each fault will be satisfied by a compression and a decompression rather than a pair of disk I/Os.

Another scenario, as mentioned above, is the application that spreads its accesses uniformly in an address space much larger than physical memory. Although taking some of memory for the CCACHE will result in additional page faults that would otherwise not occur, the average cost of a page fault will be less with the compression cache in use as long as compression is much less expensive than disk I/O (say, one-third the cost) and the hit rate in the compression cache is reasonably high (say, 80–90%).

Appendix A30
Invalidity of U.S. Patent 7,181,608 based on Douglis

Douglis 1

Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on-line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2

1 Introduction

Over the past decade, the processing power and physical memory size of typical computers have increased dramatically. Even as workstation memory sizes are increasing, however, a new technology trend is pushing toward small memories: mobile computers that are smaller than their desk-top counterparts and are typically configured with significantly less memory. Application designers are sometimes forced to squeeze their applications to fit into available memory, and may not succeed. Therefore, in a general-purpose mobile computer, as with many computers, paging is needed to enable a wider range of applications to run—as long as it can be performed efficiently.

Douglis 2

The potential benefits of the compression cache depend on the relationship between the speed of compression and the I/O bandwidth of the system, as well as the compression ratio (anywhere from barely over 1:1 to about 4:1 in the experiments reported below). If the cost of compressing and copying a page were negligible, and pages compressed well, the compression cache could be used to give a computer the appearance of having additional physical memory. In practice, compressing and copying have costs associated with them, and the benefit of reducing traffic to the backing store is offset by the overhead of the compression cache. Overhead comes not only from the compression itself but from the additional page faults an application will experience when some memory is used for compressed pages (as well as the data structures used to support compressed pages). Furthermore, as mentioned above, not all applications compress well: for poorly suited applications, the effort to compress memory will be wasted and degrade rather than improve performance. Thus, depending on the application and the hardware environment, the benefits of reduced I/O may outweigh the costs of compression and additional faults, or vice-versa. Configuring the compression cache to improve performance in the first case while staying out of the way in the second case is an interesting, and difficult, problem.

Douglis 2

Douglis Claim 8

"The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data"

9.1 maintaining a list of application data associated with an application program;	Douglis, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Douglis, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Douglis, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

3.3 Target Applications

I next address the issue of what applications might make effective use of a compression cache. Obviously, many applications will fit into memory without the need for compression, and other applications may cause excessive I/O even with compression. The applications that can best make use of a CCACHE are those that will not comfortably fit into physical memory without compression but will fit if some of their pages are compressed.

One can imagine a contrived scenario in which the CCACHE would provide significant performance benefits: if an application cycles linearly through a working set that is one page larger than the maximum number of pages allowed to be resident, and a least-recently-used algorithm is used for page replacement, then the process will take a page fault on each new page. With the compression cache, the process will still fault on each page, but each fault will be satisfied by a compression and a decompression rather than a pair of disk I/Os.

Another scenario, as mentioned above, is the application that spreads its accesses uniformly in an address space much larger than physical memory. Although taking some of memory for the CCACHE will result in additional page faults that would otherwise not occur, the average cost of a page fault will be less with the compression cache in use as long as compression is much less expensive than disk I/O (say, one-third the cost) and the hit rate in the compression cache is reasonably high (say, 80–90%).

Douglis 1

Appendix A30
Invalidity of U.S. Patent 7,181,608 based on Douglis

Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on-line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2

1 Introduction

Over the past decade, the processing power and physical memory size of typical computers have increased dramatically. Even as workstation memory sizes are increasing, however, a new technology trend is pushing toward small memories: mobile computers that are smaller than their desk-top counterparts and are typically configured with significantly less memory. Application designers are sometimes forced to squeeze their applications to fit into available memory, and may not succeed. Therefore, in a general-purpose mobile computer, as with many computers, paging is needed to enable a wider range of applications to run—as long as it can be performed efficiently.

Douglis 2

The potential benefits of the compression cache depend on the relationship between the speed of compression and the I/O bandwidth of the system, as well as the compression ratio (anywhere from barely over 1:1 to about 4:1 in the experiments reported below). If the cost of compressing and copying a page were negligible, and pages compressed well, the compression cache could be used to give a computer the appearance of having additional physical memory. In practice, compressing and copying have costs associated with them, and the benefit of reducing traffic to the backing store is offset by the overhead of the compression cache. Overhead comes not only from the compression itself but from the additional page faults an application will experience when some memory is used for compressed pages (as well as the data structures used to support compressed pages). Furthermore, as mentioned above, not all applications compress well: for poorly suited applications, the effort to compress memory will be wasted and degrade rather than improve performance. Thus, depending on the application and the hardware environment, the benefits of reduced I/O may outweigh the costs of compression and additional faults, or vice-versa. Configuring the compression cache to improve performance in the first case while staying out of the way in the second case is an interesting, and difficult, problem.

Douglis 2

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Douglis, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this limitation:

Abstract

Compression has been used in numerous ways for many years, but recently two factors have combined in a way to push compression to the forefront of distributed systems. First, the disparity between processor speeds and I/O rates is ever-increasing, making it possible to perform compression in software to a much greater extent than was previously feasible. Second, the growth of new applications demanding enormous data rates, such as digital video and audio, makes hardware compression increasingly desirable. I discuss the importance of compression in various environments and describe how compression may be used not only to reduce the demand for disk space, disk bandwidth, and network bandwidth, but also to appear to extend physical memory.

Douglis 1, Abstact

Data compression has long been used as a method to reduce the number of bits being stored or transmitted [7], but stored where and transmitted over what? The answer to this question is changing as the gap between processing speed and I/O bandwidth increases, and the uses for computers shift toward multimedia and other high-data-rate applications.

Traditionally, compression was used to reduce disk storage demands, by selectively compressing files that had not been accessed for a long time, and to reduce bandwidth requirements over such media as wide-area networks and modems. Some forms of compression are now nearly ubiquitous: for example, the vast majority of "anonymous FTP" archives store all but the most trivial of files in compressed format, saving disk space while making retrievals over the Internet faster. (Other sites sometimes store the original files in uncompressed format but allow on-the-fly compression upon request.) Images impose especially

Appendix A30
Invalidity of U.S. Patent 7,181,608 based on Douglis

Douglis 1, at 1

Even today, compression buys a significant improvement in disk capacity with only a slight degradation in performance. Cate and Gross combined compression and caching in a two-level file store, which automatically compresses least-recently-used files in order to save disk space [3]. Stacker[1] uses compression to increase the effective capacity of a PC hard disk, sometimes improving I/O performance but often being measurably slower than without compression; the slowness is due to the relative performance of an Intel 386 chip compared to a typical PC hard drive, and can be improved with a special processor board [5]. Taunton described a mechanism for compressing binary executables on Acorn personal computers, reducing disk space requirements and improving file system bandwidth; since only decompression was being performed on-line, the processing overhead was offset by the reduction in I/O [10]. Burrows, et al., combined on-line compression with Sprite LFS [9] to compress and decompress all disk I/O automatically [2]. They used it primarily to increase effective disk capacity but expect that with hardware compression, the increase in effective disk bandwidth would improve overall system performance.

Douglis 1, 2

memory can allow a user to run larger processes without buying more memory. In this case, virtual memory pages would be compressed and written back into physical memory rather than to a backing store. This idea was briefly mentioned by Appel and Li, who claimed that user-level support for compression would be more effective than generic operating system support [1]. Nevertheless, I believe that OS support for compressed virtual memory can prove useful. I refer to the technique of trading regular physical memory for data stored in compressed format as a *compression cache* (CCACHE).

One question, of course, is whether paging is even interesting or desirable. In some environments, paging is largely passé due to the large amounts of memory available [4]. However, I believe there are still systems and applications that could benefit greatly from a cheaper method of paging. Mobile computers are especially likely to have less memory available than needed, and the cost of paging over a wireless network or onto a small local disk might be much more than the cost of compressing the page. Even in a workstation-based environment, a main-memory database that is within a factor of two or three of the size of available physical memory might fit completely in memory in compressed format. The important issues are how effective compression is at saving memory and how costly compression is by comparison to network or disk I/O.

In the best case, compression can be done in hardware, and the cost of compression will be limited only by memory-to-memory bandwidth. In this case I would expect the CCACHE to provide a substantial improvement in performance. Beyond this, however, I argue that on-line compression in software can still provide benefits to a wide class of applications if designed properly. In this section I describe an implementation of the CCACHE in software, and the performance of some benchmarks, to support this claim.

Douglis 1, 2

There are a number of potential benefits of using a compression cache. First, compressing a page and retaining it in memory can be cheaper than writing it to backing store. Second, reading it back from memory is likely to be much cheaper than reading it from backing store, even taking the cost of decompression into account. Finally, if the page is eventually written to backing store, it can be written in compressed format, requiring less bandwidth.

The CCACHE has some potential disadvantages as well. Most importantly, it competes with user processes (and perhaps the file system cache) for memory. Putting a page in the CCACHE has the effect of freeing up only part of a page, so more pages must be transferred from uncompressed memory to the CCACHE in order to make room for one new page. Processes will take additional page faults because they are given less memory. (This suggests that the CCACHE should be of a variable size, and should be used only when its presence improves performance rather than degrading it.) Additionally, the CCACHE adds overhead in the form of added complexity during page faults, as well as extra code and data associated with the cache.

Douglis 1, 3.1

As was mentioned above, the CCACHE is primarily targeted for mobile computers, which typically have less memory than desk-based computers of the same generation. Nevertheless, the idea of the CCACHE extends naturally to any environment that pages when applications use more virtual memory than physical memory. I have added a compression cache to the

Sprite operating system [8], running on DECstation[2] 5000/200 workstations (approximately 18 SPECmarks, running a 25 MHz MIPS R3000 processor). Sprite is a particularly suitable system for the CCACHE, because it already supports the idea of trading physical memory between the virtual memory system and the file system [6]. Trading memory a third way, including the compression cache, would be a natural extension of this technique.

The initial prototype of the compression cache has been simplified in a number of ways. First of all, I use a fixed-size CCACHE, which is configured at kernel load time. The fixed-size cache is simpler but results in unneeded overhead for applications that would otherwise fit in memory; measurements of the effects of this are presented below. Secondly, compressed pages are written out using the same number of bytes as uncompressed pages: a full 4-Kbyte file block. In the future compressed pages will be combined into a smaller number of file blocks to save disk bandwidth. Third, I use only a single LZ-based compression algorithm [11] for all data. The compression algorithm reduces pages to 30–40% of their original size on average, depending on the application mix; I would expect application-specific compression algorithms to do much better. Finally, in order to estimate the effects of compression in a system with a large gap between processor and I/O speed, I configured the test system to page to a local disk running the original Sprite file system [6] rather than the faster Sprite LFS [9].

Douglis 1, 3.2

As a second test, I ran an application that responds to queries based on the contents of a hash table. The database containing the hash table was over 40 Mbytes, all of which was cached in memory, and the total address space of the process was over 60 Mbytes. The benchmark consisted of running a set of 9 queries against this database; the queries were run once to load the process's address space with the appropriate parts of its database and a second time to evaluate performance. The benchmark was run on the same DECstation as the previous measurements, but without the artificial restriction, so it had about 25 Mbytes of memory available. Running with a 12-Mbyte CCACHE was 32% faster than without the compression cache (9.28 minutes versus 13.7).

Appendix A30
Invalidity of U.S. Patent 7,181,608 based on Douglis

Douglis 1, 3.4

Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2, Abstract

4 Design

This section describes the design and implementation of a compression cache in Sprite. Sprite is largely compatible with 4.3 BSD UNIX, but its virtual memory system has an interesting difference from most versions of UNIX: physical memory is traded dynamically between VM for application processes and the file system's buffer cache [9]. Since the compression cache must vary in size dynamically as well, Sprite provides a good framework for prototyping the compression cache. The idea of the compression cache should extend naturally to UNIX,[3] Mach, or other systems; in fact, Mach's external pager interface [7] should be an excellent foundation for future work in this area.

The target environment for this research consists of mobile computers with limited memory and network bandwidth, and with small local disks or no disks at all. Because of the limited availability of Sprite, the compression cache has been prototyped in a workstation environment, running on DECstation 5000/200 workstations, paging to a local RZ57 disk. The Sprite kernel is configurable at boot-time to allow the system to use a variable amount of physical memory, so a 32-Mbyte machine can behave as though it has as little as 12 Mbytes. About 6 Mbytes are used by the kernel for code, page tables, and some forms of tracing that cannot currently be disabled.

Douglis 2, 4

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Douglis, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this limitation:

Abstract

Compression has been used in numerous ways for many years, but recently two factors have combined in a way to push compression to the forefront of distributed systems. First, the disparity between processor speeds and I/O rates is ever-increasing, making it possible to perform compression in software to a much greater extent than was previously feasible. Second, the growth of new applications demanding enormous data rates, such as digital video and audio, makes hardware compression increasingly desirable. I discuss the importance of compression in various environments and describe how compression may be used not only to reduce the demand for disk space, disk bandwidth, and network bandwidth, but also to appear to extend physical memory.

Douglis 1, Abstact

Data compression has long been used as a method to reduce the number of bits being stored or transmitted [7], but stored where and transmitted over what? The answer to this question is changing as the gap between processing speed and I/O bandwidth increases, and the uses for computers shift toward multimedia and other high-data-rate applications.

Traditionally, compression was used to reduce disk storage demands, by selectively compressing files that had not been accessed for a long time, and to reduce bandwidth requirements over such media as wide-area networks and modems. Some forms of compression are now nearly ubiquitous: for example, the vast majority of "anonymous FTP" archives store all but the most trivial of files in compressed format, saving disk space while making retrievals over the Internet faster. (Other sites sometimes store the original files in uncompressed format but allow on-the-fly compression upon request.) Images impose especially

Douglis 1, at 1

Even today, compression buys a significant improvement in disk capacity with only a slight degradation in performance. Cate and Gross combined compression and caching in a two-level file store, which automatically compresses least-recently-used files in order to save disk space [3]. Stacker[1] uses compression to increase the effective capacity of a PC hard disk, sometimes improving I/O performance but often being measurably slower than without compression; the slowness is due to the relative performance of an Intel 386 chip compared to a typical PC hard drive, and can be improved with a special processor board [5]. Taunton described a mechanism for compressing binary executables on Acorn personal computers, reducing disk space requirements and improving file system bandwidth; since only decompression was being performed on-line, the processing overhead was offset by the reduction in I/O [10]. Burrows, et al., combined on-line compression with Sprite LFS [9] to compress and decompress all disk I/O automatically [2]. They used it primarily to increase effective disk capacity but expect that with hardware compression, the increase in effective disk bandwidth would improve overall system performance.

Douglis 1, 2

memory can allow a user to run larger processes without buying more memory. In this case, virtual memory pages would be compressed and written back into physical memory rather than to a backing store. This idea was briefly mentioned by Appel and Li, who claimed that user-level support for compression would be more effective than generic operating system support [1]. Nevertheless, I believe that OS support for compressed virtual memory can prove useful. I refer to the technique of trading regular physical memory for data stored in compressed format as a *compression cache* (CCACHE).

One question, of course, is whether paging is even interesting or desirable. In some environments, paging is largely passé due to the large amounts of memory available [4]. However, I believe there are still systems and applications that could benefit greatly from a cheaper method of paging. Mobile computers are especially likely to have less memory available than needed, and the cost of paging over a wireless network or onto a small local disk might be much more than the cost of compressing the page. Even in a workstation-based environment, a main-memory database that is within a factor of two or three of the size of available physical memory might fit completely in memory in compressed format. The important issues are how effective compression is at saving memory and how costly compression is by comparison to network or disk I/O.

In the best case, compression can be done in hardware, and the cost of compression will be limited only by memory-to-memory bandwidth. In this case I would expect the CCACHE to provide a substantial improvement in performance. Beyond this, however, I argue that on-line compression in software can still provide benefits to a wide class of applications if designed properly. In this section I describe an implementation of the CCACHE in software, and the performance of some benchmarks, to support this claim.

Douglis 1, 2

There are a number of potential benefits of using a compression cache. First, compressing a page and retaining it in memory can be cheaper than writing it to backing store. Second, reading it back from memory is likely to be much cheaper than reading it from backing store, even taking the cost of decompression into account. Finally, if the page is eventually written to backing store, it can be written in compressed format, requiring less bandwidth.

The CCACHE has some potential disadvantages as well. Most importantly, it competes with user processes (and perhaps the file system cache) for memory. Putting a page in the CCACHE has the effect of freeing up only part of a page, so more pages must be transferred from uncompressed memory to the CCACHE in order to make room for one new page. Processes will take additional page faults because they are given less memory. (This suggests that the CCACHE should be of a variable size, and should be used only when its presence improves performance rather than degrading it.) Additionally, the CCACHE adds overhead in the form of added complexity during page faults, as well as extra code and data associated with the cache.

Douglis 1, 3.1

As was mentioned above, the CCACHE is primarily targeted for mobile computers, which typically have less memory than desk-based computers of the same generation. Nevertheless, the idea of the CCACHE extends naturally to any environment that pages when applications use more virtual memory than physical memory. I have added a compression cache to the

Sprite operating system [8], running on DECstation[2] 5000/200 workstations (approximately 18 SPECmarks, running a 25 MHz MIPS R3000 processor). Sprite is a particularly suitable system for the CCACHE, because it already supports the idea of trading physical memory between the virtual memory system and the file system [6]. Trading memory a third way, including the compression cache, would be a natural extension of this technique.

The initial prototype of the compression cache has been simplified in a number of ways. First of all, I use a fixed-size CCACHE, which is configured at kernel load time. The fixed-size cache is simpler but results in unneeded overhead for applications that would otherwise fit in memory; measurements of the effects of this are presented below. Secondly, compressed pages are written out using the same number of bytes as uncompressed pages: a full 4-Kbyte file block. In the future compressed pages will be combined into a smaller number of file blocks to save disk bandwidth. Third, I use only a single LZ-based compression algorithm [11] for all data. The compression algorithm reduces pages to 30–40% of their original size on average, depending on the application mix; I would expect application-specific compression algorithms to do much better. Finally, in order to estimate the effects of compression in a system with a large gap between processor and I/O speed, I configured the test system to page to a local disk running the original Sprite file system [6] rather than the faster Sprite LFS [9].

Douglis 1, 3.2

As a second test, I ran an application that responds to queries based on the contents of a hash table. The database containing the hash table was over 40 Mbytes, all of which was cached in memory, and the total address space of the process was over 60 Mbytes. The benchmark consisted of running a set of 9 queries against this database; the queries were run once to load the process's address space with the appropriate parts of its database and a second time to evaluate performance. The benchmark was run on the same DECstation as the previous measurements, but without the artificial restriction, so it had about 25 Mbytes of memory available. Running with a 12-Mbyte CCACHE was 32% faster than without the compression cache (9.28 minutes versus 13.7).

Douglis 1, 3.4

Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2, Abstract

4 Design

This section describes the design and implementation of a compression cache in Sprite. Sprite is largely compatible with 4.3 BSD UNIX, but its virtual memory system has an interesting difference from most versions of UNIX: physical memory is traded dynamically between VM for application processes and the file system's buffer cache [9]. Since the compression cache must vary in size dynamically as well, Sprite provides a good framework for prototyping the compression cache. The idea of the compression cache should extend naturally to UNIX,[3] Mach, or other systems; in fact, Mach's external pager interface [7] should be an excellent foundation for future work in this area.

The target environment for this research consists of mobile computers with limited memory and network bandwidth, and with small local disks or no disks at all. Because of the limited availability of Sprite, the compression cache has been prototyped in a workstation environment, running on DECstation 5000/200 workstations, paging to a local RZ57 disk. The Sprite kernel is configurable at boot-time to allow the system to use a variable amount of physical memory, so a 32-Mbyte machine can behave as though it has as little as 12 Mbytes. About 6 Mbytes are used by the kernel for code, page tables, and some forms of tracing that cannot currently be disabled.

Douglis 2, 4

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Douglis, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Douglis, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Douglis, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

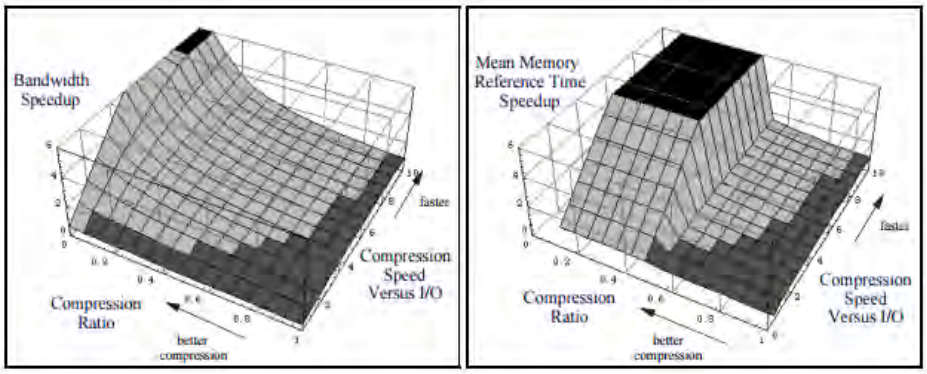
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this limitation:

See Claims 1.3 and 1.4 above.

See also

The Compression Cache . . .



(a) Transferring compressed pages to backing store. (b) Keeping compressed pages in memory.

Figure 1: Performance of compressing pages, modeled analytically. Speedups are shown as a function of the compression ratio (fraction of bytes left after compression) and the speed of compression relative to I/O. Decompression is assumed to be twice as fast as compression, as is roughly the case for algorithms such as LZRW1 [16]. There are three regions of speedup: the dark black areas at the top left show speedups that go off the top of the scale (6-fold improvement); the light areas show speedups of 1–6 relative to no compression, and the darker areas to the right show data points at which a slowdown would result.

[16] Ross N. Williams. An extremely fast ZIV-Lempel data compression algorithm. In *Data Compression Conference*, pages 362–371, April 1991.

Douglis 2

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Douglis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

See also

Burrows *et al.* integrated compression with Sprite LFS [12], also primarily to reduce disk space requirements [4]. They argued that LFS is a better vehicle for compressing files than traditional file systems, since files are not overwritten in place and a change to one block within a file would not cause changes to compressed data later in the file. Multiple file blocks may be compressed as a unit, providing better compression than if each block were compressed separately using a dynamic compression algorithm such as LZRW1 [16]. Burrows *et al.* found that on-line compression halved disk space requirements, as in Cate and Gross's system, without the delays that could be incurred by decompressing a large file as a single unit. The system had an acceptable performance degradation when compression was performed in software, and was well-suited to hardware compression.

Douglis 2

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Douglis, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Douglis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Douglis, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Douglis, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Douglis, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Douglis, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Douglis, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Douglis, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Douglis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Douglis, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Douglis, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Douglis, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Douglis, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Douglis, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Douglis, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Douglis, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Douglis, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Douglis, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 15 and 27 above	

Appendix A30
Invalidity of U.S. Patent 7,181,608 based on Douglis

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Douglis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 16 and 27 above	

Appendix A31
Invalidity of U.S. Patent 7,181,608 based on Grove

The publication Grove "System Administration," LINUX, Mar 1998. ("Grove") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A31
Invalidity of U.S. Patent 7,181,608 based on Grove

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Grove, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this claim limitation:

In order to compile the kernel, you must have the `gcc` C compiler installed on your system. `gcc` version 2.6.3 or a more recent version is required to compile the 2.0 kernel.

First `cd` to `/usr/src/linux`. The command `make config` prompts you for a number of configuration options. This is the step where you select the hardware that your kernel will support. The biggest mistake to avoid is not including support for your hard disk controller. Without the correct hard disk support in the kernel, the system won't even boot. If you are unsure about what a kernel option means, a short description is available by pressing ? and Enter.

Next, run the command `make dep` to update all of the source dependencies. This is an important step. `make clean` removes old binary files from the kernel source tree.

The command `make zImage` compiles the kernel and writes it to `/usr/src/linux/arch/i386/boot/zImage`. Linux kernels on Intel systems are always compressed. Sometimes the kernel you want to compile is too large to be compressed with the compression system that `make zImage` uses. A kernel which is too large will exit the kernel compile with the error message: `Kernel Image Too Large`. If this happens, try the command `make bzImage`, which uses a compression system that supports larger kernels. The kernel is written to `/usr/src/linux/arch/i386/boot/bzImage`.

Once you have the kernel compiled, you need to either copy it to a boot floppy (with a command like ``cp zImage /dev/fd0''") or install the image so LILO will boot from your hard drive. See page ⌴ for more information.

Grove, 4.9.1

1.1 maintaining a list of boot data used for booting a computer system;	Grove, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this claim limitation:

In order to compile the kernel, you must have the `gcc` C compiler installed on your system. `gcc` version 2.6.3 or a more recent version is required to compile the 2.0 kernel.

First `cd` to `/usr/src/linux`. The command `make config` prompts you for a number of configuration options. This is the step where you select the hardware that your kernel will support. The biggest mistake to avoid is not including support for your hard disk controller. Without the correct hard disk support in the kernel, the system won't even boot. If you are unsure about what a kernel option means, a short description is available by pressing ? and Enter.

Next, run the command `make dep` to update all of the source dependencies. This is an important step. `make clean` removes old binary files from the kernel source tree.

The command `make zImage` compiles the kernel and writes it to `/usr/src/linux/arch/i386/boot/zImage`. Linux kernels on Intel systems are always compressed. Sometimes the kernel you want to compile is too large to be compressed with the compression system that `make zImage` uses. A kernel which is too large will exit the kernel compile with the error message: `Kernel Image Too Large`. If this happens, try the command `make bzImage`, which uses a compression system that supports larger kernels. The kernel is written to `/usr/src/linux/arch/i386/boot/bzImage`.

Once you have the kernel compiled, you need to either copy it to a boot floppy (with a command like ``cp zImage /dev/fd0''") or install the image so LILO will boot from your hard drive. See page ⌴ for more information.

Grove, 4.9.1

1.2 initializing a central processing unit of the computer system;	Grove, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this claim limitation:

4.3.1 The /etc/inittab file.

Immediately after Linux boots and the kernel mounts the root file system, the first program that the system executes is `init`. This program is responsible for starting the system startup scripts, and modifies the system operating from its initial boot-up state to its standard, multiuser state. `init` also spawns the `login:` shells for all of the tty devices on the system, and specifies other startup and shutdown procedures.

After startup, `init` remains quietly in the background, monitoring and if necessary altering the running state of the system. There are many details that the `init` program must see to. These tasks are defined in the `/etc/inittab` file. A sample `/etc/inittab` file is shown below.

Grove, 4.3.1

In order to compile the kernel, you must have the `gcc` C compiler installed on your system. `gcc` version 2.6.3 or a more recent version is required to compile the 2.0 kernel.

First `cd` to `/usr/src/linux`. The command `make config` prompts you for a number of configuration options. This is the step where you select the hardware that your kernel will support. The biggest mistake to avoid is not including support for your hard disk controller. Without the correct hard disk support in the kernel, the system won't even boot. If you are unsure about what a kernel option means, a short description is available by pressing ? and Enter.

Next, run the command `make dep` to update all of the source dependencies. This is an important step. `make clean` removes old binary files from the kernel source tree.

The command `make zImage` compiles the kernel and writes it to `/usr/src/linux/arch/i386/boot/zImage`. Linux kernels on Intel systems are always compressed. Sometimes the kernel you want to compile is too large to be compressed with the compression system that `make zImage` uses. A kernel which is too large will exit the kernel compile with the error message: `Kernel Image Too Large`. If this happens, try the command `make bzImage`, which uses a compression system that supports larger kernels. The kernel is written to `/usr/src/linux/arch/i386/boot/bzImage`.

Once you have the kernel compiled, you need to either copy it to a boot floppy (with a command like "`cp zImage /dev/fd0`") or install the image so LILO will boot from your hard drive. See page _ for more information.

Grove, 4.9.1

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Grove, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this claim limitation:

In order to compile the kernel, you must have the `gcc` C compiler installed on your system. `gcc` version 2.6.3 or a more recent version is required to compile the 2.0 kernel.

First `cd` to `/usr/src/linux`. The command `make config` prompts you for a number of configuration options. This is the step where you select the hardware that your kernel will support. The biggest mistake to avoid is not including support for your hard disk controller. Without the correct hard disk support in the kernel, the system won't even boot. If you are unsure about what a kernel option means, a short description is available by pressing ? and Enter.

Next, run the command `make dep` to update all of the source dependencies. This is an important step. `make clean` removes old binary files from the kernel source tree.

The command `make zImage` compiles the kernel and writes it to `/usr/src/linux/arch/i386/boot/zImage`. Linux kernels on Intel systems are always compressed. Sometimes the kernel you want to compile is too large to be compressed with the compression system that `make zImage` uses. A kernel which is too large will exit the kernel compile with the error message: `Kernel Image Too Large`. If this happens, try the command `make bzImage`, which uses a compression system that supports larger kernels. The kernel is written to `/usr/src/linux/arch/i386/boot/bzImage`.

Once you have the kernel compiled, you need to either copy it to a boot floppy (with a command like "`cp zImage /dev/fd0`") or install the image so LILO will boot from your hard drive. See page ⌟ for more information.

Grove, 4.9.1

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Grove, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this claim limitation:

In order to compile the kernel, you must have the `gcc` C compiler installed on your system. `gcc` version 2.6.3 or a more recent version is required to compile the 2.0 kernel.

First `cd` to `/usr/src/linux`. The command `make config` prompts you for a number of configuration options. This is the step where you select the hardware that your kernel will support. The biggest mistake to avoid is not including support for your hard disk controller. Without the correct hard disk support in the kernel, the system won't even boot. If you are unsure about what a kernel option means, a short description is available by pressing ? and Enter.

Next, run the command `make dep` to update all of the source dependencies. This is an important step. `make clean` removes old binary files from the kernel source tree.

The command `make zImage` compiles the kernel and writes it to `/usr/src/linux/arch/i386/boot/zImage`. Linux kernels on Intel systems are always compressed. Sometimes the kernel you want to compile is too large to be compressed with the compression system that `make zImage` uses. A kernel which is too large will exit the kernel compile with the error message: `Kernel Image Too Large`. If this happens, try the command `make bzImage`, which uses a compression system that supports larger kernels. The kernel is written to `/usr/src/linux/arch/i386/boot/bzImage`.

Once you have the kernel compiled, you need to either copy it to a boot floppy (with a command like ``cp zImage /dev/fd0'') or install the image so LILO will boot from your hard drive. See page ⌟ for more information.

Grove, 4.9.1

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Grove, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Grove, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

Page describes how to back up files to a tape drive. Linux provides support for a variety of tape drives with IDE, SCSI, and some proprietary interfaces. Another common type of tape drive connects directly to the floppy drive controller. Linux provides the ftape device driver as a module.

At the time of this writing, the most recent version of ftape is 3.04d. You can retrieve the package from the `sunsite.unc.edu` FTP archive (see Appendix B for instructions). The ftape archive is located in `/pub/Linux/kernel/tapes`. Be sure to get the most recent version. At the time of this writing, this is `ftape-3.04d.tar.gz`.

After unpacking the ftape archive in the `/usr/src` directory, typing `make install` in the top-level ftape directory will compile the ftape driver modules and utilities, if necessary, and install them. If you experience compatibility problems with the ftape executable distribution files and your system kernel or libraries, executing the commands `make clean` and `make install` will ensure that the modules are compiled on your system.

To use this version of the ftape driver, you must have module support compiled into the kernel, as well as support for the `kerneld` kernel daemon. However, you must *not* include the kernel's built-in ftape code as a kernel option, as the more recent ftape module completely replaces this code.

Grove, 4.9.3

4. The method of claim 1, further comprising updating the list of boot data.	Grove, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this limitation:

See Claim 1.1 above.

See also

4.9.1 Upgrading the kernel

Upgrading the kernel is a matter of obtaining the kernel sources and compiling them. This is generally a painless procedure, but you can run into problems if you try to upgrade to a development kernel, or upgrade to a new kernel version. The version of a kernel has two parts, the kernel version and patchlevel. As of the time of this writing, the latest stable kernel is version `2.0.33`. The `2.0` is the kernel version and `33` is the patch level. Odd-numbered kernel versions like `2.1` are development kernels. Stay away from development kernels unless you want to live dangerously! As a general rule, you should be able to upgrade easily to another patch level, but upgrading to a new version requires the upgrade of system utilities which interact closely with the kernel.

The Linux kernel sources may be retrieved from any of the Linux FTP sites (see page ... for a list). On `sunsite.unc.edu`, for instance, the kernel sources are found in `/pub/Linux/kernel`, organized into subdirectories by version number.

Kernel sources are released as a gzipped tar file. For example, the file containing the 2.0.33 kernel sources is `linux-2.0.33.tar.gz`.

Kernel sources are unpacked in the `/usr/src` directory, creating the directory `/usr/src/linux`. It is common practice for `/usr/src/linux` to be a soft link to another directory which contains the version number, like `/usr/src/linux-2.0.33`. This way, you can install new kernel sources and test them out before removing the old kernel sources. The commands to create the kernel directory link are

```
# cd /usr/src
# mkdir linux-2.0.33
# rm -r linux
# ln -s linux-2.0.33 linux
# tar xzf linux-2.0.33.tar.gz
```

Grove, 4.9.1

In order to compile the kernel, you must have the `gcc` C compiler installed on your system. `gcc` version 2.6.3 or a more recent version is required to compile the 2.0 kernel.

First `cd` to `/usr/src/linux`. The command `make config` prompts you for a number of configuration options. This is the step where you select the hardware that your kernel will support. The biggest mistake to avoid is not including support for your hard disk controller. Without the correct hard disk support in the kernel, the system won't even boot. If you are unsure about what a kernel option means, a short description is available by pressing ? and Enter.

Next, run the command `make dep` to update all of the source dependencies. This is an important step. `make clean` removes old binary files from the kernel source tree.

The command `make zImage` compiles the kernel and writes it to `/usr/src/linux/arch/i386/boot/zImage`. Linux kernels on Intel systems are always compressed. Sometimes the kernel you want to compile is too large to be compressed with the compression system that `make zImage` uses. A kernel which is too large will exit the kernel compile with the error message: `Kernel Image Too Large`. If this happens, try the command `make bzImage`, which uses a compression system that supports larger kernels. The kernel is written to `/usr/src/linux/arch/i386/boot/bzImage`.

Once you have the kernel compiled, you need to either copy it to a boot floppy (with a command like ``cp zImage /dev/fd0'') or install the image so LILO will boot from your hard drive. See page ⌟ for more information.

Grove, 4.9.1

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Grove, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Grove, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Grove, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Grove, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

Page describes how to back up files to a tape drive. Linux provides support for a variety of tape drives with IDE, SCSI, and some proprietary interfaces. Another common type of tape drive connects directly to the floppy drive controller. Linux provides the ftape device driver as a module.

At the time of this writing, the most recent version of ftape is 3.04d. You can retrieve the package from the `sunsite.unc.edu` FTP archive (see Appendix B for instructions). The ftape archive is located in `/pub/Linux/kernel/tapes`. Be sure to get the most recent version. At the time of this writing, this is `ftape-3.04d.tar.gz`.

After unpacking the ftape archive in the `/usr/src` directory, typing `make install` in the top-level ftape directory will compile the ftape driver modules and utilities, if necessary, and install them. If you experience compatibility problems with the ftape executable distribution files and your system kernel or libraries, executing the commands `make clean` and `make install` will ensure that the modules are compiled on your system.

To use this version of the ftape driver, you must have module support compiled into the kernel, as well as support for the `kerneld` kernel daemon. However, you must *not* include the kernel's built-in ftape code as a kernel option, as the more recent ftape module completely replaces this code.

Grove, 4.9.3

7.2 a cache memory device; and;	Grove, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Grove, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Grove, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Grove, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Grove, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.	

9.1 maintaining a list of application data associated with an application program;	Grove, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Grove, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Grove, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.4, 2, and 8 above.	

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Grove, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this limitation:

In order to compile the kernel, you must have the `gcc` C compiler installed on your system. `gcc` version 2.6.3 or a more recent version is required to compile the 2.0 kernel.

First `cd` to `/usr/src/linux`. The command `make config` prompts you for a number of configuration options. This is the step where you select the hardware that your kernel will support. The biggest mistake to avoid is not including support for your hard disk controller. Without the correct hard disk support in the kernel, the system won't even boot. If you are unsure about what a kernel option means, a short description is available by pressing ? and Enter.

Next, run the command `make dep` to update all of the source dependencies. This is an important step. `make clean` removes old binary files from the kernel source tree.

The command `make zImage` compiles the kernel and writes it to `/usr/src/linux/arch/i386/boot/zImage`. Linux kernels on Intel systems are always compressed. Sometimes the kernel you want to compile is too large to be compressed with the compression system that `make zImage` uses. A kernel which is too large will exit the kernel compile with the error message: `Kernel Image Too Large`. If this happens, try the command `make bzImage`, which uses a compression system that supports larger kernels. The kernel is written to `/usr/src/linux/arch/i386/boot/bzImage`.

Once you have the kernel compiled, you need to either copy it to a boot floppy (with a command like ``cp zImage /dev/fd0'') or install the image so LILO will boot from your hard drive. See page ⌟ for more information.

Grove, 4.9.1

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Grove, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this limitation:

In order to compile the kernel, you must have the `gcc` C compiler installed on your system. `gcc` version 2.6.3 or a more recent version is required to compile the 2.0 kernel.

First `cd` to `/usr/src/linux`. The command `make config` prompts you for a number of configuration options. This is the step where you select the hardware that your kernel will support. The biggest mistake to avoid is not including support for your hard disk controller. Without the correct hard disk support in the kernel, the system won't even boot. If you are unsure about what a kernel option means, a short description is available by pressing ? and Enter.

Next, run the command `make dep` to update all of the source dependencies. This is an important step. `make clean` removes old binary files from the kernel source tree.

The command `make zImage` compiles the kernel and writes it to `/usr/src/linux/arch/i386/boot/zImage`. Linux kernels on Intel systems are always compressed. Sometimes the kernel you want to compile is too large to be compressed with the compression system that `make zImage` uses. A kernel which is too large will exit the kernel compile with the error message: `Kernel Image Too Large`. If this happens, try the command `make bzImage`, which uses a compression system that supports larger kernels. The kernel is written to `/usr/src/linux/arch/i386/boot/bzImage`.

Once you have the kernel compiled, you need to either copy it to a boot floppy (with a command like ``cp zImage /dev/fd0'') or install the image so LILO will boot from your hard drive. See page ⌟ for more information.

Grove, 4.9.1

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Grove, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Grove, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Grove, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Grove, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.3, 1.4, and 15 above.	

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Grove, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Grove, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above	

22.1 maintaining a list of boot data used for booting a computer system;.	Grove, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claim 1.1 above	

22.2 initializing a central processing unit of the computer system;.	Grove, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claim 1.2 above	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Grove, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claim 1.3 above	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Grove, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claim 1.4 above	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Grove, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 4, 10 and 11 above	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Grove, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 15 and 22 above	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Grove, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 16 and 22 above	

27.1 a boot device..	Grove, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claim 1 above	

27.2 a processor..	Grove, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claim 1.2 above	

27.3 cache memory; and.	Grove, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.3 and 1.4 above	

27.4 non-volatile memory for storing logic code for use by the processor,..	Grove, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claim 1.1 and 1.3 above	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Grove, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claim 1.1 above	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Grove, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claim 1.3 above	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Grove, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claim 1.4 above	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Grove, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 4, 10, and 11 above	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Grove, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 15 and 27 above	

Appendix A31
Invalidity of U.S. Patent 7,181,608 based on Grove

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Grove, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 16 and 27 above	

Appendix A32
Invalidity of U.S. Patent 7,181,608 based on Jones

The publication Jones, The Microsoft Interactive TV system: An Experience Report, July 1997 ("Jones") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Jones, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

> "The MMOSA real-time kernel was developed as a cooperative effort between members of the operating systems research group within Microsoft Research and a team of development staff committed to building a small, embedded-systems kernel meeting the needs of interactive television. MMOSA is the internal name for the ITV kernel: it stands for MultiMedia Operating System Architecture."

Jones, §4.

> "Set-top boxes boot as follows. First, a small boot loader in EPROM is run. This decompresses a boot image also in the ROM and jumps to its entry point. The ROM boot image contains the MMOSA kernel, the network protocol stack and drivers, and a network boot program. The network boot program sends a DHCP request. It waits for a DHCP reply containing not only the usual IP address, but also some MITV-specific extended results such as sets of IP addresses of Class Store servers and a boot filename to use. It then loads a new boot image from a Class Store via TFTP and jumps to it."

Jones, 14 at § 10.

1.1 maintaining a list of boot data used for booting a computer system;	Jones, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

> "The MMOSA real-time kernel was developed as a cooperative effort between members of the operating systems research group within Microsoft Research and a team of development staff committed to building a small, embedded-systems kernel meeting the needs of interactive television. MMOSA is the internal name for the ITV kernel: it stands for MultiMedia Operating System Architecture."

Jones, §4.

> "Set-top boxes boot as follows. First, a small boot loader in EPROM is run. This decompresses a boot image also in the ROM and jumps to its entry point. The ROM boot image contains the MMOSA kernel, the network protocol stack and drivers, and a network boot program. The network boot program sends a DHCP request. It waits for a DHCP reply containing not only the usual IP address, but also some MITV-specific extended results such as sets of IP addresses of Class Store servers and a boot filename to use. It then loads a new boot image from a Class Store via TFTP and jumps to it."

Jones, 14 at § 10.

1.2 initializing a central processing unit of the computer system;	Jones, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Jones, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

> "The processor has an 8Ki/8Kd on-chip cache."

Jones, § 3.

> "Set-top boxes boot as follows. First, a small boot loader in EPROM is run. This decompresses a boot image also in the ROM and jumps to its entry point. The ROM boot image contains the MMOSA kernel, the network protocol stack and drivers, and a network boot program. The network boot program sends a DHCP request. It waits for a DHCP reply containing not only the usual IP address, but also some MITV-specific extended results such as sets of IP addresses of Class Store servers and a boot filename to use. It then loads a new boot image from a Class Store via TFTP and jumps to it."

Jones, § 10.

> "Kanji font not in ROM. The set-top boxes we built for the NTT trial contained 1/2 MB of EPROM. This was sufficient for storing the compressed boot image but wasn't large enough to hold a Kanji font."

Jones, § 15.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Jones, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

> "Set-top boxes boot as follows. First, a small boot loader in EPROM is run. This decompresses a boot image also in the ROM and jumps to its entry point. The ROM boot image contains the MMOSA kernel, the network protocol stack and drivers, and a network boot program. The network boot program sends a DHCP request. It waits for a DHCP reply containing not only the usual IP address, but also some MITV-specific extended results such as sets of IP addresses of Class Store servers and a boot filename to use. It then loads a new boot image from a Class Store via TFTP and jumps to it."

Jones, 14 at § 10.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Jones, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "The system is designed to be able to provide traditional television service, video-on-demand service, and custom interactive applications, such as interactive shopping services, to subscriber bases scaling from a few hundred subscribers up through major metropolitan areas."

Jones, Abstract.

> "Open software platform supporting old, new, and unforeseen applications and third-party development."

Jones, §1.

> "From March, 1996 to April, 1997 this system served 298 paying subscribers, providing the following applications to users:
> • Movies on demand. The same code is used with different data to provide movies-on-demand, sports-on-demand, animation-on-demand (cartoons), and cooking-on-demand services.
> • Electronic program guide -- an interactive TV guide to available broadcast channels.
> • Viewing standard broadcast television channels.
> • NTT customer service application. Provides subscriber logon, channel lockout, billing queries, etc.

Applications written and deployed by NTT include:
• Karaoke on demand.
• On-line shopping services."

Jones, §2.

"DCOM is used to implement many application services, including program invocation, remote database services (used for program and movie guide data), per-viewer and per-set-top-box persistent state (such as profile data and last channel watched), financial transaction services, and logging. Finally, on top of these lower layers, the actual user-visible applications, such as the video-on-demand player, the digital broadcast player, the electronic program guide, and the system navigator (the channel surfing application), are built."

Jones, §2.1.

"Approximately 12 megabytes of general-purpose set-top box memory are in use when running typical applications, such as video-on-demand."

Jones, §2.1.

"Application Portability: Applications written to the Win32 subset could be easily ported between the set-top box and existing Windows platforms. For instance, some games and Internet Explorer were both ported to the set-top box environment."

Jones, §6.1.

"The namespace is used by client code for locating and connecting to a number of potentially replicated services at boot time and application start time."

Jones, §6.4.

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Jones, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

> "For instance the Tiger controller server runs on a dedicated machine, and the Tiger cubs (the actual server machines from which striped video is delivered) occupy ten dedicated server machines, leaving four other general head-end server machines in this configuration."

Jones, §2.2. *See also* Jones, Fig 2-1.

> "The set-top box also has a bi-directional infrared port for communicating with the hand controller, a smart card interface, a serial port (used for debugging), a microphone input, auxiliary audio & video inputs, and separate audio and video outputs for TV and VCR."

Jones, §3.

> "The Tiger system deployed in the NTT trial uses one dedicated Tiger controller server and ten dedicated Tiger Cub servers (the machines across which the data are striped)."

Jones, §8.1.

4. The method of claim 1, further comprising updating the list of boot data.	Jones, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claim 1.1 above.	

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Jones, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Jones, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.1 and 4 above.	

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Jones, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Jones, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

> "For instance the Tiger controller server runs on a dedicated machine, and the Tiger cubs (the actual server machines from which striped video is delivered) occupy ten dedicated server machines, leaving four other general head-end server machines in this configuration."

Jones, §2.2. *See also* Jones, Fig 2-1.

> "The set-top box also has a bi-directional infrared port for communicating with the hand controller, a smart card interface, a serial port (used for debugging), a microphone input, auxiliary audio & video inputs, and separate audio and video outputs for TV and VCR."

Jones, §3.

> "The Tiger system deployed in the NTT trial uses one dedicated Tiger controller server and ten dedicated Tiger Cub servers (the machines across which the data are striped)."

Jones, §8.1.

7.2 a cache memory device; and;	Jones, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Jones, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Jones, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Jones, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Jones, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> "The system is designed to be able to provide traditional television service, video-on-demand service, and custom interactive applications, such as interactive shopping services, to subscriber bases scaling from a few hundred subscribers up through major metropolitan areas."

Jones, Abstract.

> "Open software platform supporting old, new, and unforeseen applications and third-party development."

Jones, §1.

> "From March, 1996 to April, 1997 this system served 298 paying subscribers, providing the following applications to users:

> • Movies on demand. The same code is used with different data to provide movies-on-demand, sports-on-demand, animation-on-demand (cartoons), and cooking-on-demand services.
> • Electronic program guide -- an interactive TV guide to available broadcast channels.
> • Viewing standard broadcast television channels.
> • NTT customer service application. Provides subscriber logon, channel lockout, billing queries, etc.
>
> Applications written and deployed by NTT include:
> • Karaoke on demand.
> • On-line shopping services."

Jones, §2.

> "DCOM is used to implement many application services, including program invocation, remote database services (used for program and movie guide data), per-viewer and per-set-top-box persistent state (such as profile data and last channel watched), financial transaction services, and logging. Finally, on top of these lower layers, the actual user-visible applications, such as the video-on-demand player, the digital broadcast player, the electronic program guide, and the system navigator (the channel surfing application), are built."

Jones, §2.1.

> "Approximately 12 megabytes of general-purpose set-top box memory are in use when running typical applications, such as video-on-demand."

Jones, §2.1.

> "Application Portability: Applications written to the Win32 subset could be easily ported between the set-top box and existing Windows platforms. For instance, some games and Internet Explorer were both ported to the set-top box environment."

Jones, §6.1.

> "The namespace is used by client code for locating and connecting to a number of potentially replicated services at boot time and application start time."

Jones, §6.4.

9.1 maintaining a list of application data associated with an application program;	Jones, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Jones, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Jones, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> "The system is designed to be able to provide traditional television service, video-on-demand service, and custom interactive applications, such as interactive shopping services, to subscriber bases scaling from a few hundred subscribers up through major metropolitan areas."

Jones, Abstract.

> "Open software platform supporting old, new, and unforeseen applications and third-party development."

Jones, §1.

> "From March, 1996 to April, 1997 this system served 298 paying subscribers, providing the following applications to users:

• Movies on demand. The same code is used with different data to provide movies-on-demand, sports-on-demand, animation-on-demand (cartoons), and cooking-on-demand services.
• Electronic program guide -- an interactive TV guide to available broadcast channels.
• Viewing standard broadcast television channels.
• NTT customer service application. Provides subscriber logon, channel lockout, billing queries, etc.

Applications written and deployed by NTT include:
• Karaoke on demand.
• On-line shopping services."

Jones, §2.

"DCOM is used to implement many application services, including program invocation, remote database services (used for program and movie guide data), per-viewer and per-set-top-box persistent state (such as profile data and last channel watched), financial transaction services, and logging. Finally, on top of these lower layers, the actual user-visible applications, such as the video-on-demand player, the digital broadcast player, the electronic program guide, and the system navigator (the channel surfing application), are built."

Jones, §2.1.

"Approximately 12 megabytes of general-purpose set-top box memory are in use when running typical applications, such as video-on-demand."

Jones, §2.1.

"Application Portability: Applications written to the Win32 subset could be easily ported between the set-top box and existing Windows platforms. For instance, some games and Internet Explorer were both ported to the set-top box environment."

Jones, §6.1.

"The namespace is used by client code for locating and connecting to a number of potentially replicated services at boot time and application start time."

Jones, §6.4.

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Jones, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

> "The MMOSA real-time kernel was developed as a cooperative effort between members of the operating systems research group within Microsoft Research and a team of development staff committed to building a small, embedded-systems kernel meeting the needs of interactive television. MMOSA is the internal name for the ITV kernel: it stands for MultiMedia Operating System Architecture."

Jones, §4.

> "Set-top boxes boot as follows. First, a small boot loader in EPROM is run. This decompresses a boot image also in the ROM and jumps to its entry point. The ROM boot image contains the MMOSA kernel, the network protocol stack and drivers, and a network boot program. The network boot program sends a DHCP request. It waits for a DHCP reply containing not only the usual IP address, but also some MITV-specific extended results such as sets of IP addresses of Class Store servers and a boot filename to use. It then loads a new boot image from a Class Store via TFTP and jumps to it."

Jones, § 10.

> "Kanji font not in ROM. The set-top boxes we built for the NTT trial contained 1/2 MB of EPROM. This was sufficient for storing the compressed boot image but wasn't large enough to hold a Kanji font."

Jones, § 15.

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Jones, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

> "Set-top boxes boot as follows. First, a small boot loader in EPROM is run. This decompresses a boot image also in the ROM and jumps to its entry point. The ROM boot image contains the MMOSA kernel, the network protocol stack and drivers, and a network boot program. The network boot program sends a DHCP request. It waits for a DHCP reply containing not only the usual IP address, but also some MITV-specific extended results such as sets of IP addresses of Class Store servers and a boot filename to use. It then loads a new boot image from a Class Store via TFTP and jumps to it."

Jones, 14 at § 10.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Jones, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Jones, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Jones, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.3 and 1.4 above.	

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Jones, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

See also

> "Custom video and audio hardware for the set-top box contains a MPEG-2 decoder, NTSC (the U.S. and Japanese analog television encoding standard) encoders & decoders, a tuner, and an audio mixer."

Jones, §3.

> "Everything from server machines and ATM switches to real-time MPEG-2 encoders and OS software was precisely duplicated."

Jones, §11.

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Jones, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Jones, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above.	

22.1 maintaining a list of boot data used for booting a computer system;.	Jones, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claim 1.1 above.	

22.2 initializing a central processing unit of the computer system;.	Jones, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claim 1.2 above.	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Jones, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claim 1.3 above.	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Jones, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claim 1.4 above.	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Jones, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 4, 10 and 11 above.	

24. The method of claim 22, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Jones, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 15 and 22 above.	

25. The method of claim 22, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data..	Jones, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 16 and 22 above.	

Appendix A32
Invalidity of U.S. Patent 7,181,608 based on Jones

27.1 a boot device..	Jones, as evidenced by the example citations below, discloses "a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claim 1 above.	

27.2 a processor..	Jones, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claim 1.2 above.	

27.3 cache memory; and.	Jones, as evidenced by the example citations below, discloses "cache memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, cache memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.3 and 1.4 above.	

27.4 non-volatile memory for storing logic code for use by the processor,..	Jones, as evidenced by the example citations below, discloses "non-volatile memory for storing logic code for use by the processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, non-volatile memory for storing logic code for use by the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claim 1.1 and 1.3 above.	

27.5 the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system	Jones, as evidenced by the example citations below, discloses "the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the logic code being used for: maintaining a list associated with boot data, wherein the boot data is used in booting a first system;), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claim 1.1 above.	

27.6 preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system; and	Jones, as evidenced by the example citations below, discloses "preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading compressed boot data associated to the list into the cache memory prior to completion of initialization of a central processing unit of the first system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claim 1.3 above.	

27.7 servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit; and	Jones, as evidenced by the example citations below, discloses "servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for the compressed boot data from the first system after completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claim 1.4 above.	

27.8 a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device	Jones, as evidenced by the example citations below, discloses "a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for decompressing the compressed boot data accessed from the cache memory for use in responding to the servicing requests and for compressing additional boot data and storing the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 4, 10, and 11 above.	

29. The system of claim 27, wherein Lempel-Ziv is utilized by the data compression engine to compress the additional boot data.	Jones, as evidenced by the example citations below, discloses "Lempel-Ziv is utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv is utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 15 and 27 above.	

30. The system of claim 27, wherein a plurality of encoders are utilized by the data compression engine to compress the additional boot data.	Jones, as evidenced by the example citations below, discloses "a plurality of encoders are utilized by the data compression engine to compress the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized by the data compression engine to compress the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 16 and 27 above.	

Appendix A33
Invalidity of U.S. Patent 7,181,608 based on Magstar

The publication Magstar and IBM 3590 High Performance Tape Subsystem Technical Guide, November 1996 ("Magstar") invalidates claims 1-13, 15-16, 19-20, 22, 24-25, 27, and 29-30 of United States Patent No. 7,181,608 ("the '608 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '608 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix A33
Invalidity of U.S. Patent 7,181,608 based on Magstar

1 (Preamble) A method for providing accelerated loading of an operating system, comprising the steps of:	Magstar, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system, comprising the steps of:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

> "The drive data rate at 9 MB/s is three times faster than the IBM 3480, 3490, or 3490E tape drives. This increase in drive data rate, together with the built-in data compression capability, makes it possible to utilize more effectively the full capability of a 20 MB/s fast-and-wide SCSI or a 17 MB/s ESCON channel."

Magstar at 25.

> "If the compression ratio achieved is greater than 3:1, the attainable throughput is constrained by the channel speed. Nevertheless, for most applications using conventional DASD, the data rate attained is determined not by the tape drive speed but by another system or application component."

Magstar, 133.

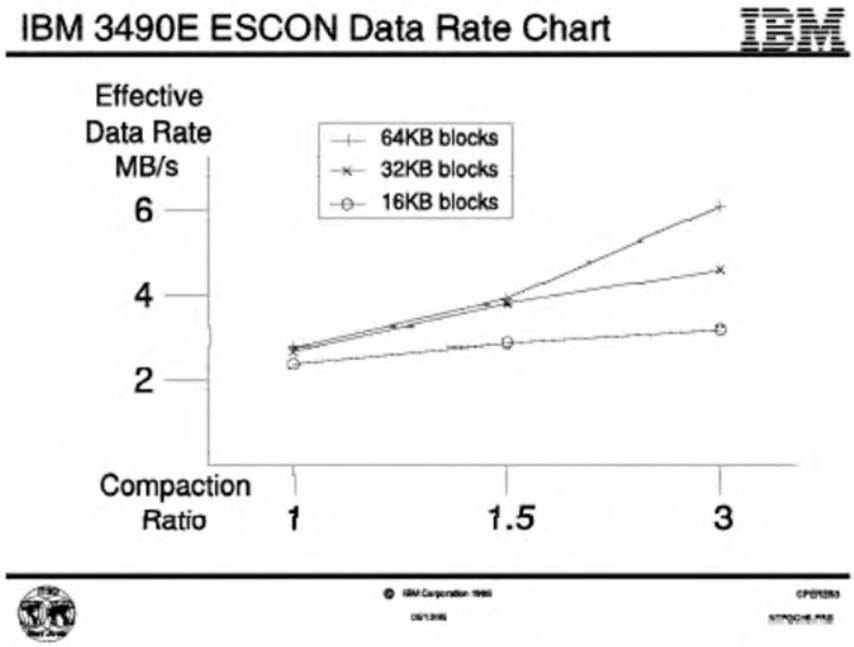
> "Two data patterns were used in this study to provide compaction ratios of approximately 1.5:1 and 3:1. A third set of results for the uncompacted data case was achieved by setting the JCL parameter TRTCH=NOCOMP to turn off the IDRC compression. Table 8 on page 134, taken from WSC Flash 9108 IBM 3490E Tape Subsystem Performance, which documents the study, relates to a subsystem such as the subsystem shown in the diagram, using an ESCON channel, an 8MB buffer, and with five drives active."

Magstar, 133.

Table 8. Effective Data Rates per Path to an IBM 3490E Tape Drive

Compaction Ratio	16KB Blocks (MB/s)	32KB Blocks (MB/s)	64KB Blocks (MB/s)
No compaction	2.4	2.7	2.8
1.5:1	2.9	3.8	3.9
3:1	3.2	4.6	6.1

Magstar, Table 8.



Magstar, Figure 60.

> "The measured data rates illustrate two points about effective data rates as opposed to native data rates in tape subsystems:
>
> - When the data is not compressed, the effective data rate is limited by the drive speed, that is, approximately 3MB/s. However, when the data is compressed, the maximum data rate is approximately 6MB/s.
> - The data rate is dependent on block size. With a block size of 16KB and a compression ratio of 3:1, the maximum data rate is 3.2MB/s."

Magstar, 135. *See also generally*, Magstar 135-145.

> "New longitudinal technology now announced by IBM (16-track

Serpentine Interleaved Longitudinal Recording) significantly improves tape performance and transfer rates without changing the tape speed (2 m/s). A buffer is used and the data compressed before it is written to." Magstar, 12.

1.1 maintaining a list of boot data used for booting a computer system;	Magstar, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.2 initializing a central processing unit of the computer system;	Magstar, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.3 preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and	Magstar, as evidenced by the example citations below, discloses "preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the boot data into a cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the boot data comprises accessing compressed boot data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

> "The IBMLZ1 compression algorithm used in the IBM 3590 is based on the Ziv-Lempel algorithm2 and Jackson's[3] class of encoding methods."

Magstar, 25. *See also generally* Magstar, 25-26 at "Improved compression."

> "It is during the process of transferring data from the host channel to the buffer that the data can be compressed using the BAC algorithm component of IDRC."

Magstar, 132.

> "The compression achieved using IDRC is made up of two factors:
>
> - Automatic reblocking of the data to a block size of 128KB, which has a more marked effect on small block sizes
> - Applying the BAC algorithm to the data, the effectiveness of which depends on the randomness of the data, and whether or not it has already been compressed (for example, by hardware or software data compression in the host)."

Magstar, 132. *See also generally* Magstar, 132 at §6.1.2 and §6.9.

> "The IBM Magstar Virtual Tape Server Controller, Tape Volume Cache, and the IBM 3590 Magstar tape drives, together with the required

> housing, make up the IBM Magstar Virtual Tape Server (VTS) subsystem, allowing automatic utilization of the Magstar cartridge storage capacity and the drive data rate of 9 MB/s."

Magstar, 150. *See also generally* Magstar, 150-211.

> "New longitudinal technology now announced by IBM (16-track Serpentine Interleaved Longitudinal Recording) significantly improves tape performance and transfer rates without changing the tape speed (2 m/s). A buffer is used and the data compressed before it is written to."

Magstar, 12.

1.4 servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache.	Magstar, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded boot data after completion of the initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed boot data from the cache and decompressing the compressed boot data at a rate that increases the effective access rate of the cache), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

> "The drive data rate at 9 MB/s is three times faster than the IBM 3480, 3490, or 3490E tape drives. This increase in drive data rate, together with the built-in data compression capability, makes it possible to utilize more effectively the full capability of a 20 MB/s fast-and-wide SCSI or a 17 MB/s ESCON channel."

Magstar at 25.

> "The IBMLZ1 compression algorithm used in the IBM 3590 is based on the Ziv-Lempel algorithm2 and Jackson's[3] class of encoding methods."

Magstar, 25. *See also generally* Magstar, 25-26 at "Improved compression."

> "If the compression ratio achieved is greater than 3:1, the attainable throughput is constrained by the channel speed. Nevertheless, for most applications using conventional DASD, the data rate attained is determined not by the tape drive speed but by another system or application component."

Magstar, 133.

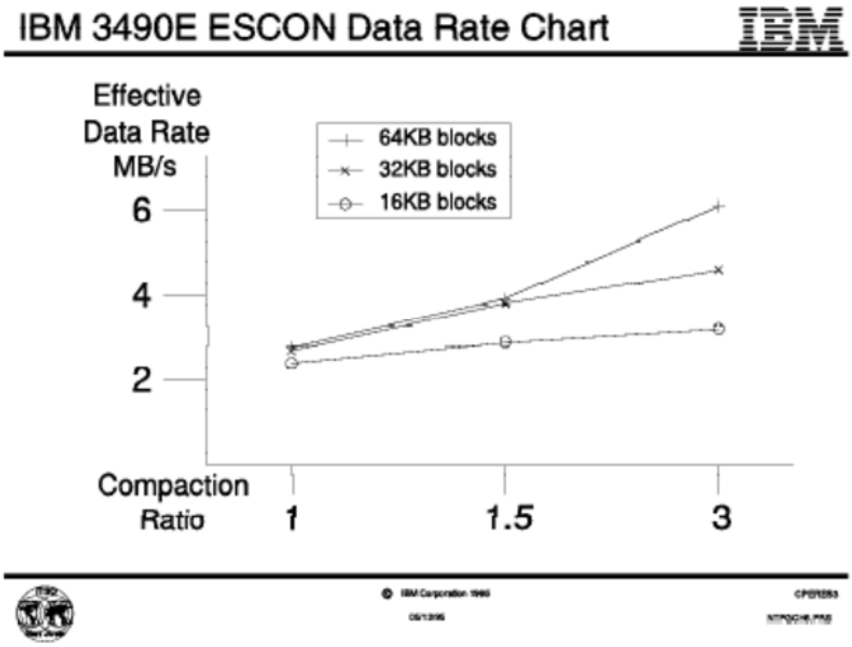
> "Two data patterns were used in this study to provide compaction ratios of approximately 1.5:1 and 3:1. A third set of results for the uncompacted data case was achieved by setting the JCL parameter TRTCH=NOCOMP to turn off the IDRC compression. Table 8 on page 134, taken from WSC Flash 9108 IBM 3490E Tape Subsystem Performance, which documents the study, relates to a subsystem such as the subsystem shown in the diagram, using an ESCON channel, an 8MB buffer, and with five drives active."

Magstar, 133.

Table 8. Effective Data Rates per Path to an IBM 3490E Tape Drive

Compaction Ratio	16KB Blocks (MB/s)	32KB Blocks (MB/s)	64KB Blocks (MB/s)
No compaction	2.4	2.7	2.8
1.5:1	2.9	3.8	3.9
3:1	3.2	4.6	6.1

Magstar, Table 8.



Magstar, Figure 60.

> "The measured data rates illustrate two points about effective data rates

as opposed to native data rates in tape subsystems:

- When the data is not compressed, the effective data rate is limited by the drive speed, that is, approximately 3MB/s. However, when the data is compressed, the maximum data rate is approximately 6MB/s.
- The data rate is dependent on block size. With a block size of 16KB and a compression ratio of 3:1, the maximum data rate is 3.2MB/s."

Magstar, 135. *See also generally*, Magstar 135-145

> "The IBM Magstar Virtual Tape Server Controller, Tape Volume Cache, and the IBM 3590 Magstar tape drives, together with the required housing, make up the IBM Magstar Virtual Tape Server (VTS) subsystem, allowing automatic utilization of the Magstar cartridge storage capacity and the drive data rate of 9 MB/s."

Magstar, 150. *See also generally* Magstar, 150-211.

> "New longitudinal technology now announced by IBM (16-track Serpentine Interleaved Longitudinal Recording) significantly improves tape performance and transfer rates without changing the tape speed (2 m/s). A buffer is used and the data compressed before it is written to."

Magstar, 12.

2. The method of claim 1, wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof.	Magstar, as evidenced by the example citations below, discloses "wherein the boot data comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "This support is functionally transparent to the user programs with the exception of those applications that use the NOTE TYPE=ABS macro and calculate the logical block number on the basis of the values returned in register 0 and resister 1. These application programs may have to be modified."

Magstar, 105.

> "Typically the user application program itself does not use the tape drive directly."

Magstar, 122.

3. The method of claim 1, wherein the preloading is performed by a data storage controller connected to the boot device.	Magstar, as evidenced by the example citations below, discloses "wherein the preloading is performed by a data storage controller connected to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the preloading is performed by a data storage controller connected to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

See Claims 1.3, and 1.4 above.

See also

> "To use the Magstar tape drive from an ES/9000 or S/390 system, an IBM 3590-A00 tape controller is required to convert the channel commands to corresponding SCSI commands."

Magstar, 96.

> "The IBM Magstar Virtual Tape Server Controller, Tape Volume Cache, and the IBM 3590 Magstar tape drives, together with the required housing, make up the IBM Magstar Virtual Tape Server (VTS) subsystem, allowing automatic utilization of the Magstar cartridge storage capacity and the drive data rate of 9 MB/s."

Magstar, 150.

> "The IBM Magstar Tape Server Controller and its associated microcode, are the key components of the Virtual Tape Server subsystem:
>
> • It automatically fills and manages Magstar 3590 cartridge capacity.
>
> • It controls and manages the volume movement to and from the tape volume cache and 3590 cartridges or Magstar tape drives."

Magstar, 150. *See also generally*, Magstar 1-269.

4. The method of claim 1, further comprising updating the list of boot data.	Magstar, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claim 1.1 above.	

5. The method of claim 4, wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list.	Magstar, as evidenced by the example citations below, discloses "wherein the step of updating comprises adding to the list any boot data requested by the computer system not previously stored in the list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, boot data that comprises program code associated with one of an operating system of the computer system, an application program, and a combination thereof), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1 and 4 above.	

6. The method of claim 4, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system.	Magstar, as evidenced by the example citations below, discloses "wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the step of updating comprises removing from the list any boot data previously stored in the list and not requested by the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.1 and 4 above.	

Appendix A33
Invalidity of U.S. Patent 7,181,608 based on Magstar

7. (Preamble) A system for providing accelerated loading of an operating system of a host system comprising:	Magstar, as evidenced by the example citations below, discloses "a system for providing accelerated loading of an operating system of a host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system for providing accelerated loading of an operating system of a host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1 (Preamble) above.	

7.1 a digital signal processor (DSP) or controller;	Magstar, as evidenced by the example citations below, discloses "a digital signal processor (DSP) or controller."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a digital signal processor (DSP) or controller), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

See Claims 1.2, 1.3, and 1.4 above.

See also

> "To use the Magstar tape drive from an ES/9000 or S/390 system, an IBM 3590-A00 tape controller is required to convert the channel commands to corresponding SCSI commands."

Magstar, 96.

> "The IBM Magstar Virtual Tape Server Controller, Tape Volume Cache, and the IBM 3590 Magstar tape drives, together with the required housing, make up the IBM Magstar Virtual Tape Server (VTS) subsystem, allowing automatic utilization of the Magstar cartridge storage capacity and the drive data rate of 9 MB/s."

Magstar, 150.

> "The IBM Magstar Tape Server Controller and its associated microcode, are the key components of the Virtual Tape Server subsystem:
>
> • It automatically fills and manages Magstar 3590 cartridge capacity.
>
> • It controls and manages the volume movement to and from the tape volume cache and 3590 cartridges or Magstar tape drives."

Magstar, 150. *See also generally*, Magstar 1-269.

7.2 a cache memory device; and;	Magstar, as evidenced by the example citations below, discloses "a cache memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a cache memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

7.3.1 a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system	Magstar, as evidenced by the example citations below, discloses "a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device, for storing logic code associated with the DSP or controller, wherein the logic code comprises instructions executable by the DSP or controller for maintaining a list of boot data used for booting the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.1, 1.3, 2, 3, and 7.1 above.	

7.3.2 for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system	Magstar, as evidenced by the example citations below, discloses "for preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the compressed boot data into the cache memory device prior to completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.3, and 1.4 above.	

7.3.3 and for decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system	Magstar, as evidenced by the example citations below, discloses "decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the preloaded compressed boot data, at a rate that increases the effective access rate of the cache, to service requests for boot data from the host system after completion of initialization of the central processing unit of the host system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.3, and 1.4 above.	

8. The system of claim 7, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data.	Magstar, as evidenced by the example citations below, discloses "wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the logic code in the non-volatile memory device further comprises program instructions executable by the DSP or controller for maintaining a list of application data associated with an application program; preloading the application data upon launching the application program, and servicing requests for the application data from the host system using the preloaded application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

See Claims 1.1, 1.3, 1.4, 2, 3, and 7 above.

See also

> "This support is functionally transparent to the user programs with the exception of those applications that use the NOTE TYPE=ABS macro and calculate the logical block number on the basis of the values returned in register 0 and resister 1. These application programs may have to be modified."

Magstar, 105.

> "Typically the user application program itself does not use the tape drive directly."

Magstar, 122.

9.1 maintaining a list of application data associated with an application program;	Magstar, as evidenced by the example citations below, discloses "maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.1, 2, and 8 above.	

9.2 preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device; and	Magstar, as evidenced by the example citations below, discloses "preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading the application data into the cache memory prior to completion of initialization of the central processing unit of the computer system, wherein preloading the application data comprises accessing compressed application data from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.3, 2, and 8 above.	

9.3 servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.	Magstar, as evidenced by the example citations below, discloses "servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data.".

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for application data from the computer system using the preloaded application data after completion of initialization of the central processing unit of the computer system, wherein servicing requests comprises accessing compressed application data from the cache and decompressing the compressed application data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

See Claims 1.4, 2, and 8 above.

See also

> "This support is functionally transparent to the user programs with the exception of those applications that use the NOTE TYPE=ABS macro and calculate the logical block number on the basis of the values returned in register 0 and resister 1. These application programs may have to be modified."

Magstar, 105.

> "Typically the user application program itself does not use the tape drive directly."

Magstar, 122.

10. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device.	Magstar, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data and the data compression engine provides the compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

> "The IBMLZ1 compression algorithm used in the IBM 3590 is based on the Ziv-Lempel algorithm2 and Jackson's[3] class of encoding methods."

Magstar, 25. *See also generally* Magstar, 25-26 at "Improved compression."

> "It is during the process of transferring data from the host channel to the buffer that the data can be compressed using the BAC algorithm component of IDRC."

Magstar, 132.

> "The compression achieved using IDRC is made up of two factors:
>
> - Automatic reblocking of the data to a block size of 128KB, which has a more marked effect on small block sizes
> - Applying the BAC algorithm to the data, the effectiveness of which depends on the randomness of the data, and whether or not it has already been compressed (for example, by hardware or software data compression in the host)."

Magstar, 132. *See also generally* Magstar, 132 at § 6.1.2. and §6.9.

> "New longitudinal technology now announced by IBM (16-track Serpentine Interleaved Longitudinal Recording) significantly improves tape performance and transfer rates without changing the tape speed (2

m/s). A buffer is used and the data compressed before it is written to." Magstar, 12.

Appendix A33
Invalidity of U.S. Patent 7,181,608 based on Magstar

11. The method of claim 1, wherein the decompressing is provided by a data compression engine.	Magstar, as evidenced by the example citations below, discloses "decompressing is provided by a data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing is provided by a data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

> "The IBMLZ1 compression algorithm used in the IBM 3590 is based on the Ziv-Lempel algorithm2 and Jackson's[3] class of encoding methods."

Magstar, 25. *See also generally* Magstar, 25-26 at "Improved compression."

> "New longitudinal technology now announced by IBM (16-track Serpentine Interleaved Longitudinal Recording) significantly improves tape performance and transfer rates without changing the tape speed (2 m/s). A buffer is used and the data compressed before it is written to."

Magstar, 12.

12. The method of claim 1, further comprising a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine.	Magstar, as evidenced by the example citations below, discloses "a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for compressing, wherein the compressing provides the compressed boot data, the data compression engine provides the compressed boot data to the boot device, and the decompressing is provided by the data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 10 and 11 above.	

13. The method of claim 1, wherein the compressed boot data is accessed via direct memory access.	Magstar, as evidenced by the example citations below, discloses "the compressed boot data is accessed via direct memory access."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the compressed boot data is accessed via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.3 and 1.4 above.	

15. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Magstar, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

See Claims 1.3 and 1.4 above.

See also

> "The Magstar tape drive also has an improved compression algorithm (Ziv-Lempel) which is called IBMLZ1 and will be more efficient than the binary arithmetic compression (BAC) algorithm used in the IBM 3480 and 3490 Tape Subsystem□s IDRC."

Magstar, 7.

> "The IBMLZ1 compression algorithm used in the IBM 3590 is based on the Ziv-Lempel algorithm2 and Jackson□s3 class of encoding methods. For the IBM 3590 IBMLZ1 compression, a 1024 bytes of history buffer is used. This implementation differs from the IDRC used in the IBM 3490E, which is based on BAC. The IBMLZ1 algorithm is expected to be more effective than IDRC."

Magstar, 25.

> "The IBMLZ1 compression algorithm is designed for robust and highly efficient compression. Key design objectives for the IBMLZ1 algorithm are:
>
> - Hardware execution efficiency: the hardware architecture should use as few machine cycles as possible to compress or decompress a byte. The architecture should maintain low complexity and use silicon technology effectively. In addition, the smallest number of machine cycles for each byte (this number is called CPB) should be used to compress or decompress data.
>
> - Robust compression: achieve good coding efficiency for broad

applications.

- Minimum system integration overhead: the maximum benefit from compression is achieved when the compression can be performed without performance loss. So the algorithm is capable of running at channel speed (20 MB/s for SCSI)."

Magstar 25-26.

16. The method of claim 1, wherein a plurality of encoders are utilized to provide the compressed boot data.	Magstar, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

See Claims 1.3, 1.4, and 15 above.

See also

> "The IBMLZ1 compression algorithm used in the IBM 3590 is based on the Ziv-Lempel algorithm2 and Jackson's3 class of encoding methods."

Magstar, 25.

19. The method of claim 7, wherein Lempel-Ziv encoding is utilized to provide the compressed boot data..	Magstar, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 15 above.	

20. The method of claim 7, wherein a plurality of encoders are utilized to provide the compressed boot data.	Magstar, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.3 and 1.4, 7, and 16 above.	

22.1 maintaining a list of boot data used for booting a computer system;.	Magstar, as evidenced by the example citations below, discloses "maintaining a list of boot data used for booting a computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claim 1.1 above.	

22.2 initializing a central processing unit of the computer system;.	Magstar, as evidenced by the example citations below, discloses "initializing a central processing unit of the computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claim 1.2 above.	

22.3 preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit;.	Magstar, as evidenced by the example citations below, discloses "preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading boot data in compressed form, based on the list of boot data, from a boot device into a cache memory prior to completion of initialization of the central processing unit), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claim 1.3 above.	

22.4 servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data	Magstar, as evidenced by the example citations below, discloses "servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing requests for boot data from the computer system using the preloaded compressed boot data after completion of initialization of the central processing unit, wherein servicing requests comprises accessing the compressed boot data from the cache and decompressing the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claim 1.4 above.	

22.5 with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device.	Magstar, as evidenced by the example citations below, discloses "with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, with a data compression engine and the data compression engine being operable to compress additional boot data and store the additional compressed boot data to the boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 4, 10 and 11 above.	