PATENT OWNER REALTIME DATA DEMONSTRATIVE EXHIBITS

Apple Inc. v. Realtime Data, LLC d/b/a/ IXO
Cases IPR2016-01737, -01738, -01739
U.S. Patent No. 8,880,862

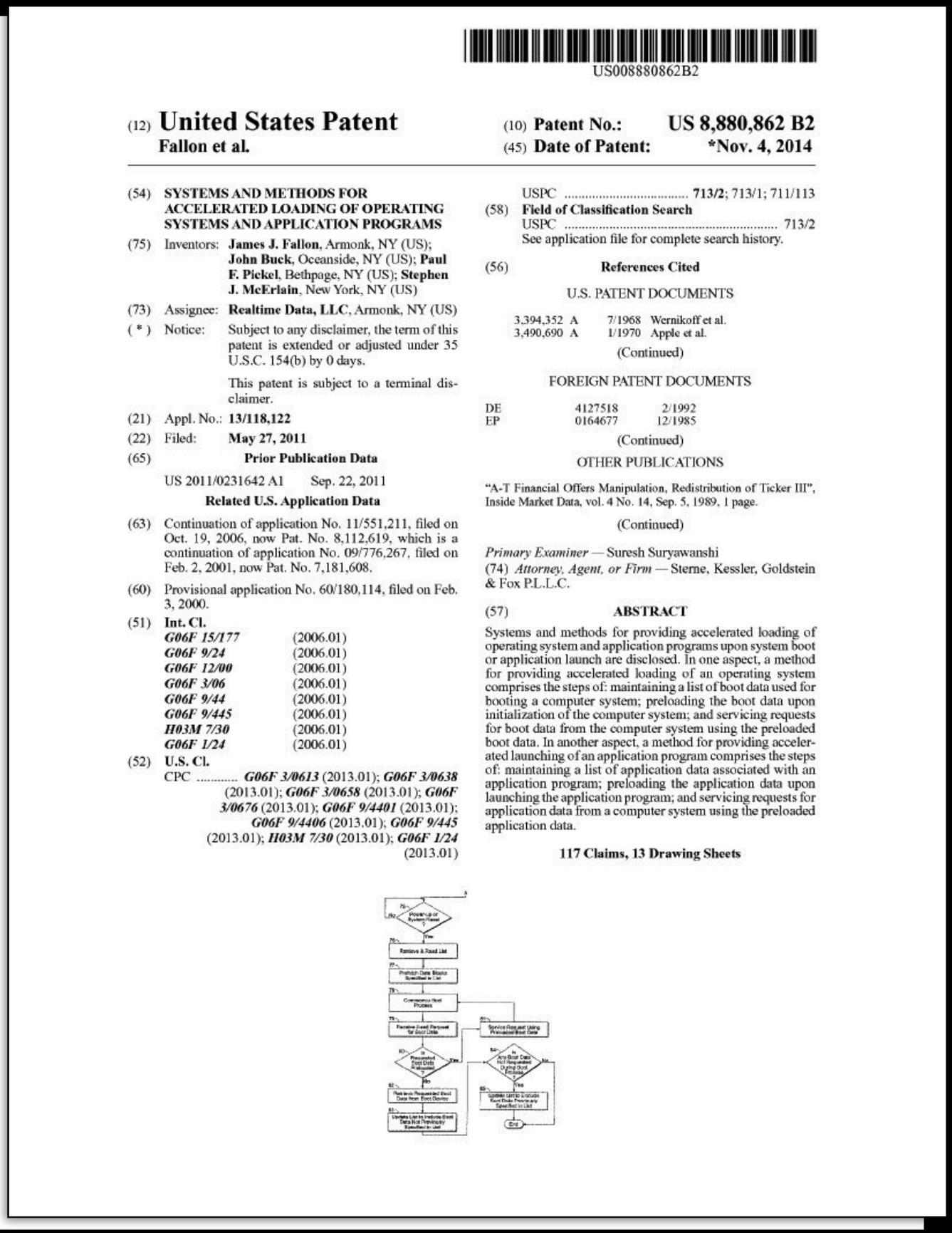
Table of Contents

IPR2016-01739 ISSUES

Table of Contents

Table of Contents

IPR2016-01737, -01738, -01739 ISSUES



US008880862B2

(12) **United States Patent**
Fallon et al.

(10) **Patent No.: US 8,880,862 B2**
(45) **Date of Patent: *Nov. 4, 2014**

(54) **SYSTEMS AND METHODS FOR ACCELERATED LOADING OF OPERATING SYSTEMS AND APPLICATION PROGRAMS**

(75) Inventors: **James J. Fallon**, Armonk, NY (US); **John Buck**, Oceanside, NY (US); **Paul F. Pickel**, Bethpage, NY (US); **Stephen J. McErlain**, New York, NY (US)

(73) Assignee: **Realtime Data, LLC**, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **13/118,122**

(22) Filed: **May 27, 2011**

(65) **Prior Publication Data**

US 2011/0231642 A1 Sep. 22, 2011

Related U.S. Application Data

(63) Continuation of application No. 11/551,211, filed on Oct. 19, 2006, now Pat. No. 8,112,619, which is a continuation of application No. 09/776,267, filed on Feb. 2, 2001, now Pat. No. 7,181,608.

(60) Provisional application No. 60/180,114, filed on Feb. 3, 2000.

(51) **Int. Cl.**

G06F 15/177	(2006.01)
G06F 9/24	(2006.01)
G06F 12/00	(2006.01)
G06F 3/06	(2006.01)
G06F 9/44	(2006.01)
G06F 9/445	(2006.01)
H03M 7/30	(2006.01)
G06F 1/24	(2006.01)

(52) **U.S. Cl.**

CPC ***G06F 3/0613*** (2013.01); ***G06F 3/0638*** (2013.01); ***G06F 3/0658*** (2013.01); ***G06F 3/0676*** (2013.01); ***G06F 9/4401*** (2013.01); ***G06F 9/4406*** (2013.01); ***G06F 9/445*** (2013.01); ***H03M 7/30*** (2013.01); ***G06F 1/24*** (2013.01)

USPC **713/2**; 713/1; 711/113

(58) **Field of Classification Search**

USPC 713/2

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,394,352 A 7/1968 Wernikoff et al.
3,490,690 A 1/1970 Apple et al.

(Continued)

FOREIGN PATENT DOCUMENTS

DE 4127518 2/1992
EP 0164677 12/1985

(Continued)

OTHER PUBLICATIONS

"A-T Financial Offers Manipulation, Redistribution of Ticker III", Inside Market Data, vol. 4 No. 14, Sep. 5, 1989, 1 page.

(Continued)

Primary Examiner — Suresh Suryawanshi

(74) *Attorney, Agent, or Firm* — Sterne, Kessler, Goldstein & Fox P.L.L.C.

(57) **ABSTRACT**

Systems and methods for providing accelerated loading of operating system and application programs upon system boot or application launch are disclosed. In one aspect, a method for providing accelerated loading of an operating system comprises the steps of: maintaining a list of boot data used for booting a computer system; preloading the boot data upon initialization of the computer system; and servicing requests for boot data from the computer system using the preloaded boot data. In another aspect, a method for providing accelerated launching of an application program comprises the steps of: maintaining a list of application data associated with an application program; preloading the application data upon launching the application program; and servicing requests for application data from a computer system using the preloaded application data.

117 Claims, 13 Drawing Sheets

IPR2016-01737

Independent Claims 1, 6, 13

Dependent Claims 3-4, 7, 23-34, 47-58, 83-96, 99-100, 105-111, 113, 116

Motion to Amend

IPR2016-01738

Independent Claims 8, 11, 14

Dependent Claims 9-10, 15-22, 59-82, 101-104, 114-115, 117

Motion to Amend

IPR2016-01739

Independent Claim 5

Dependent Claims 35-46, 97-98, 112

No Motion to Amend

"BOOT DATA LIST" CONSTRUCTION

11. A method for providing accelerated loading of an operating system in a computer system, comprising:

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;

accessing the loaded boot data in compressed form from the memory;

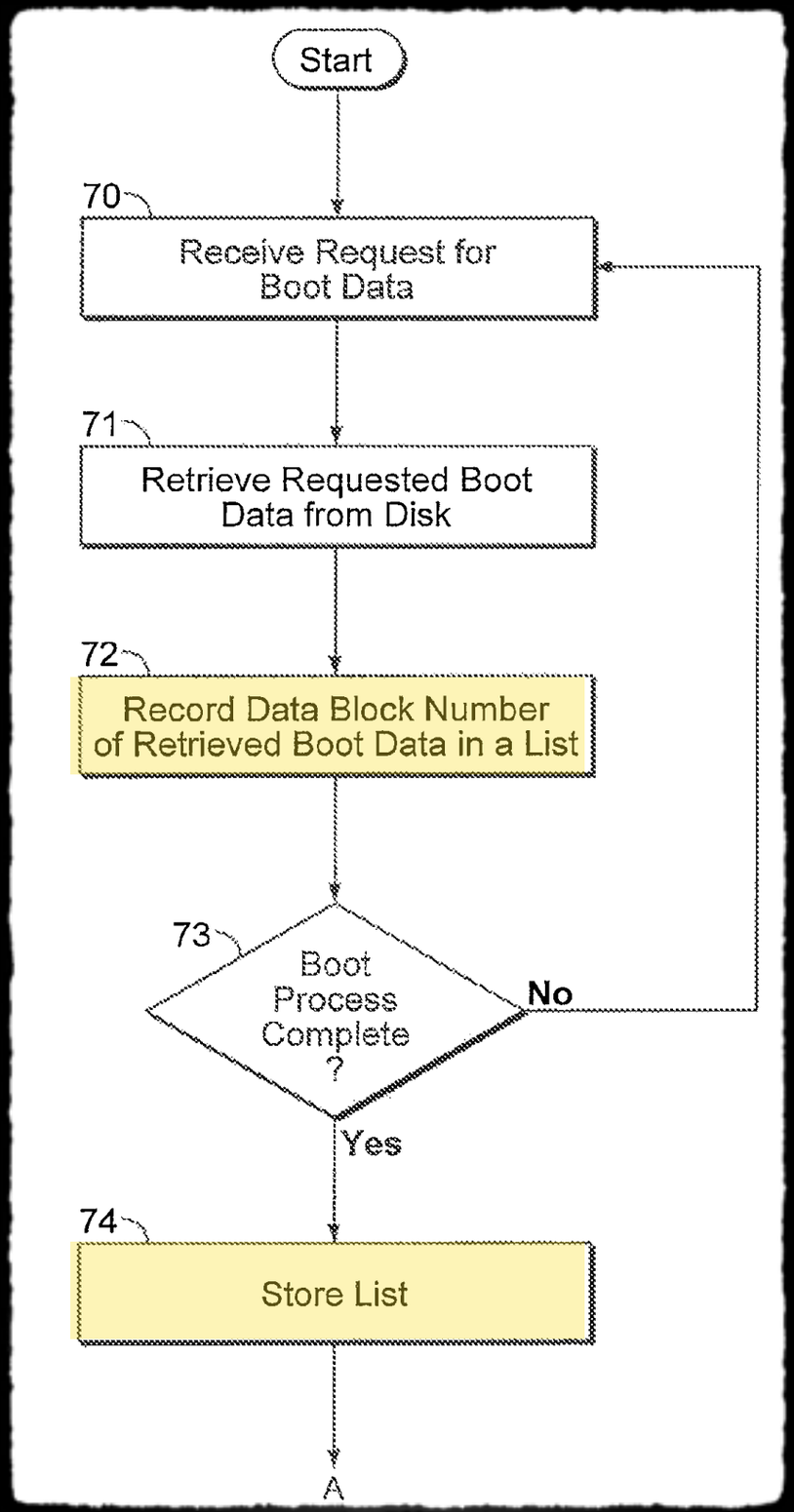
decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;

utilizing the decompressed boot data to load at least a portion of the operating system for the computer system; and

updating the boot data list.

'862 Patent at Claim 11

‘862 SPECIFICATION DISCLOSES “BOOT DATA LIST” USED TO IDENTIFY AND LOAD BOOT DATA

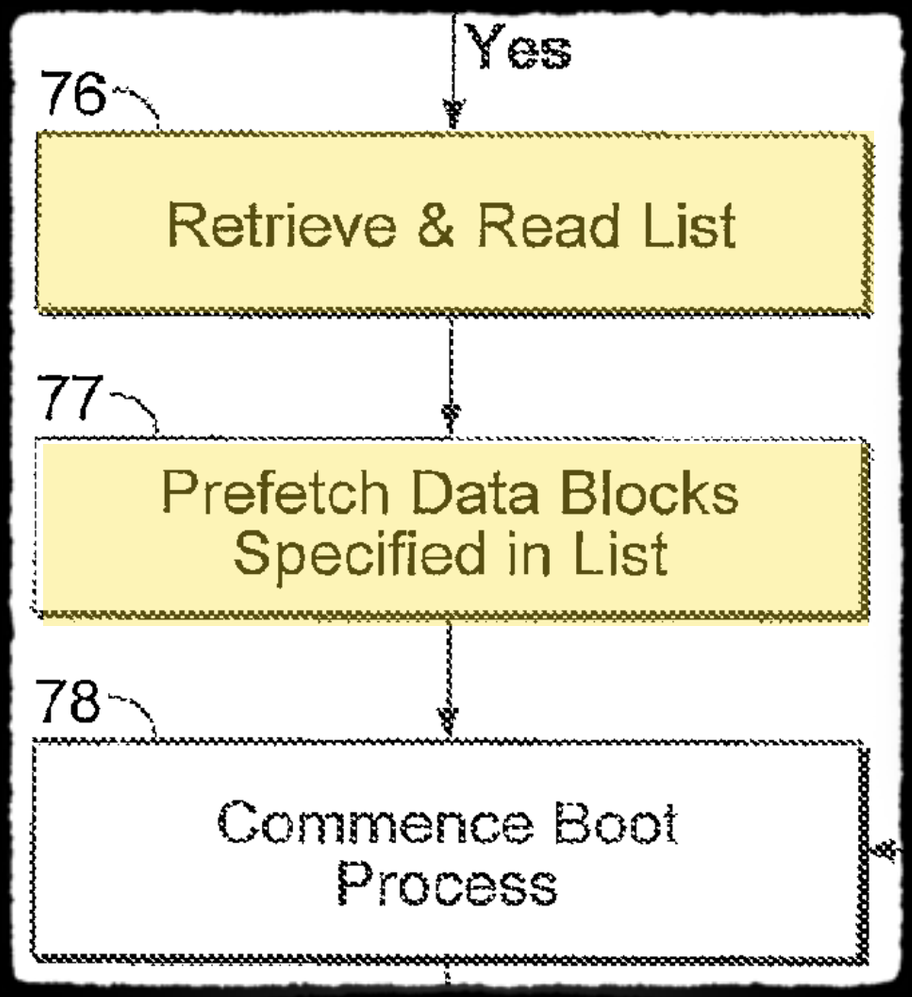


‘862 at Figure 7A

Another technique (illustrated by the flow diagram of FIGS. **7***a* and **7***b*) that may be employed comprises an automatic process that requires no input from the user. With this technique, the data storage controller maintain a list comprising the data associated with the first series of data requests received by the data storage controller by the host system after a power-on/reset. In particular, referring to FIG. **7***a*, during the computer boot process, the data storage controller will receive requests for the boot data (step **70**). In response, the data storage controller will retrieve the requested boot data from the boot device (e.g., hard disk) in the local cache memory (step **71**). For each requested data block, the data storage controller will record the requested data block number in a list (step **72**). The data storage controller will record the data block number of each data block requested by the host computer during the boot process (repeat steps **70-72**). When the boot process is complete (affirmative determination in step **73**), the data storage controller will store the data list on the boot device (or other storage device) (step **74**).

‘862 at 21:24-42

‘862 SPECIFICATION DISCLOSES “BOOT DATA LIST” USED TO IDENTIFY AND LOAD BOOT DATA



‘862 at Figure 7B

Then, upon each subsequent power-on/reset (affirmative result in step **75**), the data storage controller would retrieve and read the stored list (step **76**) and proceed to preload the boot data specified on the list (i.e., the data associated with the expected data requests) into the onboard cache memory (step **77**). It is to be understood that the depending on the resources of the given system (e.g., memory, etc.), the preloading process may be completed prior to commencement of the boot process, or continued after the boot process begins (in which case booting and preloading are performed simultaneously).

‘862 at 21:43-52

‘862 SPECIFICATION DISCLOSES “BOOT DATA LIST” USED TO IDENTIFY AND LOAD BOOT DATA

In addition, the data storage controller would update the boot data list by recording any changes in the actual data requests as compared to the expected data requests already stored in the list (step **83**).

‘862 at 21:65-22:1

POSITA WOULD UNDERSTAND "BOOT DATA LIST" IDENTIFIES AND LOADS BOOT DATA INTO MEMORY

56. Specifically, based on the '862 Patent, a POSITA would have understood that the recited "boot data list" refers to a record used to identify and load boot data into memory. As such, a POSITA would have understood that this is a record of boot data separate from the boot data itself. Notably, the specification

Dr. Back '1739 Declaration (Ex. 2008) at ¶ 56

"BOOT DATA LIST" MEANS "RECORD USED TO IDENTIFY AND LOAD BOOT DATA INTO MEMORY"

A. Proper Interpretation of "Boot Data List"

The term "boot data list," as used in claims 1-9, 11-14, 19-21, 95-106, and 111-117, should mean "record used to identify and load boot data into memory." Indeed, this construction is consistent with the claims and the intrinsic record, and is the broadest reasonable interpretation in light of the specification.

Realtime '1739 Response at 15

DR. NEUHAUSER ADMITS THAT "BOOT DATA LIST" USED TO IDENTIFY BOOT DATA TO LOAD

Q. Based on the teachings of the '862 patent, for something to constitute a boot data list, does it have to allow the data storage controller to know what data to preload from the boot device?

MR. HUGUENIN-LOVE: Object to form.

THE WITNESS: Yes, I think it does in some way.

Dr. Neuhauser Transcript (Ex. 2024) at 87:9-15

APPLE'S "BOOT DATA LIST" CONSTRUCTION IS DIVORCED FROM THE '862 SPECIFICATION AND UNREASONABLY BROAD

APPLE ARGUES THAT EVERY OPERATING SYSTEM FILE IS ITSELF A "BOOT DATA LIST":

13:55-65, FIG. 12. Because each electronic file includes a list of data stored within the file, a POSITA would have understood that an OS functional module file stored on boot device 3 includes a list of data necessary for starting the OS – a boot data list as described by the '862 Patent. Dec., ¶78 (citing Microsoft Press Computer Dictionary defining "file" as "[a] complete, named collection of information, such as a program, a set of data used by a program, or a user-created document" that "binds a conglomeration of instructions … into a coherent unit").

'1739 Petition at 22

APPLE'S "BOOT DATA LIST" CONSTRUCTION IS DIVORCED FROM SPECIFICATION AND UNREASONABLY BROAD

POR, 25-27. However, Apple argued that Settsu's files are themselves lists of boot data and Realtime does not respond to this analysis.

Specifically, Realtime criticizes the definition of "file" cited by Apple, yet offers an alternative definition that aligns with Apple's definition and argument. Specifically, Apple's definition confirmed that a file is a "collection of information." APPLE-1014, 3. Similarly, Realtime's definition states that "[a] file is a collection of related information." REALTIME-2012, 4. As Dr. Neuhauser explained, a list is an obvious representation for a collection of information and, thus, Settsu's OS files represent lists of boot data. APPLE-1003, ¶¶77-79; APPLE-1031, 5:16-20;

Apple's '1739 Reply at 9

‘862 CLAIMS AND SPECIFICATION DISTINGUISH BETWEEN “BOOT DATA” AND “BOOT DATA LIST”

What is claimed is:

1. A method for providing accelerated loading of an operating system in a computer system, the method comprising:

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 the 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; and

updating the boot data list,

wherein the decompressed portion of boot data comprises a portion of the operating system.

‘862 Patent at Claim 1

CLAIM ELEMENTS LISTED SEPARATELY IN CLAIM ARE DISTINCT COMPONENTS OF THE PATENTED INVENTION

FEDERAL CIRCUIT'S *BECTON, DICKINSON* DECISION:

"Where a claim lists elements separately, 'the clear implication of the claim language' is that those elements are 'distinct component[s]' of the patented invention."

616 F.3d 1249, 1254 (Fed. Cir. 2010); *see also HTC Corp. v. Cellular Comm'ns Equip., LLC,* IPR2014-01133, Paper 48 at 8-12 (PTAB Jan. 4, 2016)

APPLE'S "BOOT DATA LIST" CONSTRUCTION IS INCONSISTENT WITH POSITA'S UNDERSTANDING

74. Nowhere in this definition does it state that a "file" includes a list of its contents. To be sure, the term "file" is routinely defined without regard to its contents, and there is no requirement that files include lists of their contents.[48] Indeed, a POSITA would have understood that files do not necessarily contain a list of their contents. More importantly, a POSITA would have understood that the files to which Dr. Neuhauser refers in Settsu—object files created when the OS is built or rebuilt—do not contain a list of their contents. For instance, OS kernels are built in a way that eliminates the need for lists when loading the kernel into memory. The addition and use of such lists would slow down their loading. The kernel's object module contain only a header that says where to load it.[49] A POSITA would not consider a header—a single entity—to constitute a list.

Dr. Back '1739 Declaration (Ex. 2008) at ¶ 74

APPLE'S "BOOT DATA LIST" CONSTRUCTION IS INCONSISTENT WITH POSITA'S UNDERSTANDING

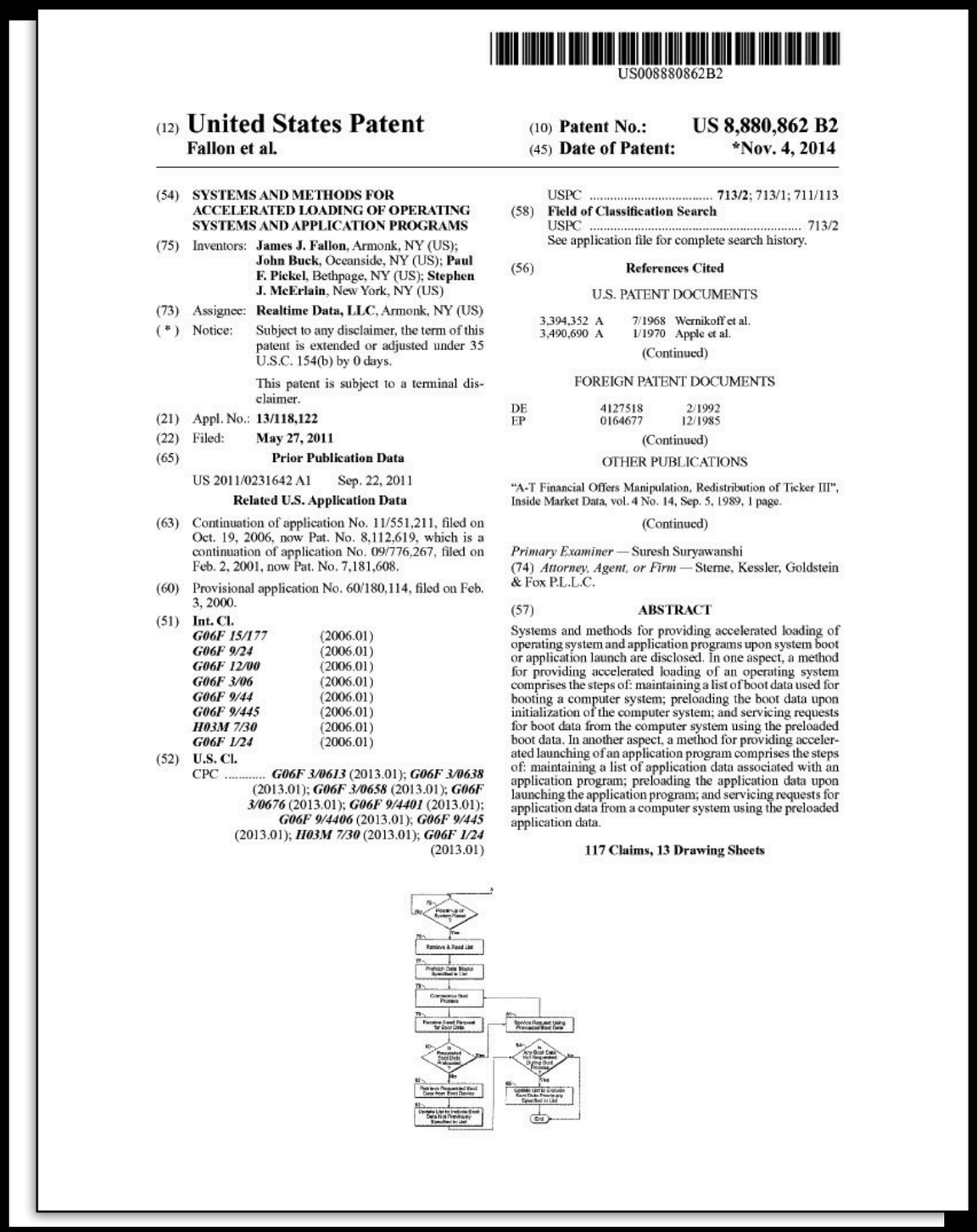
76. Moreover, Dr. Neuhauser's assertion is inconsistent with how a POSITA would have understood how a file relates to a "boot data list" and is an unreasonably broad interpretation of this claim term. As described above in Section VI.B., "boot data list" means "record used to identify and load boot data into memory." Based on this meaning, a POSITA would have understood that any information listed internally to a file of "boot data" is not a record used to identify boot data, and therefore, is not a "boot data list."

Dr. Back '1739 Declaration (Ex. 2008) at ¶ 76

IPR2016-01739 ISSUE

SETTSU AND ZWIEGINCEW DO NOT RENDER OBVIOUS "UPDATING THE BOOT DATA LIST"

CLAIMS 5, 35-46, 97, 98, AND 112 ARE PATENTABLE OVER ‘1739 IPR GROUNDS 1-4



US008880862B2

(12) **United States Patent**
Fallon et al.

(10) **Patent No.: US 8,880,862 B2**
(45) **Date of Patent: *Nov. 4, 2014**

(54) **SYSTEMS AND METHODS FOR ACCELERATED LOADING OF OPERATING SYSTEMS AND APPLICATION PROGRAMS**

(75) Inventors: **James J. Fallon**, Armonk, NY (US); **John Buck**, Oceanside, NY (US); **Paul F. Pickel**, Bethpage, NY (US); **Stephen J. McErlain**, New York, NY (US)

(73) Assignee: **Realtime Data, LLC**, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **13/118,122**

(22) Filed: **May 27, 2011**

(65) **Prior Publication Data**

US 2011/0231642 A1 Sep. 22, 2011

Related U.S. Application Data

(63) Continuation of application No. 11/551,211, filed on Oct. 19, 2006, now Pat. No. 8,112,619, which is a continuation of application No. 09/776,267, filed on Feb. 2, 2001, now Pat. No. 7,181,608.

(60) Provisional application No. 60/180,114, filed on Feb. 3, 2000.

(51) **Int. Cl.**

G06F 15/177	(2006.01)
G06F 9/24	(2006.01)
G06F 12/00	(2006.01)
G06F 3/06	(2006.01)
G06F 9/44	(2006.01)
G06F 9/445	(2006.01)
H03M 7/30	(2006.01)
G06F 1/24	(2006.01)

(52) **U.S. Cl.**

CPC ***G06F 3/0613*** (2013.01); ***G06F 3/0638*** (2013.01); ***G06F 3/0658*** (2013.01); ***G06F 3/0676*** (2013.01); ***G06F 9/4401*** (2013.01); ***G06F 9/4406*** (2013.01); ***G06F 9/445*** (2013.01); ***H03M 7/30*** (2013.01); ***G06F 1/24*** (2013.01)

USPC **713/2**; 713/1; 711/113

(58) **Field of Classification Search**

USPC 713/2

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,394,352 A 7/1968 Wernikoff et al.
3,490,690 A 1/1970 Apple et al.

(Continued)

FOREIGN PATENT DOCUMENTS

DE 4127518 2/1992
EP 0164677 12/1985

(Continued)

OTHER PUBLICATIONS

"A-T Financial Offers Manipulation, Redistribution of Ticker III", Inside Market Data, vol. 4 No. 14, Sep. 5, 1989, 1 page.

(Continued)

Primary Examiner — Suresh Suryawanshi

(74) *Attorney, Agent, or Firm* — Sterne, Kessler, Goldstein & Fox P.L.L.C.

(57) **ABSTRACT**

Systems and methods for providing accelerated loading of operating system and application programs upon system boot or application launch are disclosed. In one aspect, a method for providing accelerated loading of an operating system comprises the steps of: maintaining a list of boot data used for booting a computer system; preloading the boot data upon initialization of the computer system; and servicing requests for boot data from the computer system using the preloaded boot data. In another aspect, a method for providing accelerated launching of an application program comprises the steps of: maintaining a list of application data associated with an application program; preloading the application data upon launching the application program; and servicing requests for application data from a computer system using the preloaded application data.

117 Claims, 13 Drawing Sheets

Settsu, alone or in view of Zwiegincew, fails to render obvious claim elements:

- *“updating the boot data list" (cl. 5)*
- *“updating the boot data list in response to the utilizing" (cl. 112)*
- *“disassociating non-accessed boot data from the boot data list" (cl. 98)*

CLAIM 5'S "UPDATING THE BOOT DATA LIST" ('1739 IPR)

5. A method for booting a computer system, the method comprising:

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;

accessing the loaded compressed boot data;

decompressing the accessed compressed boot data;

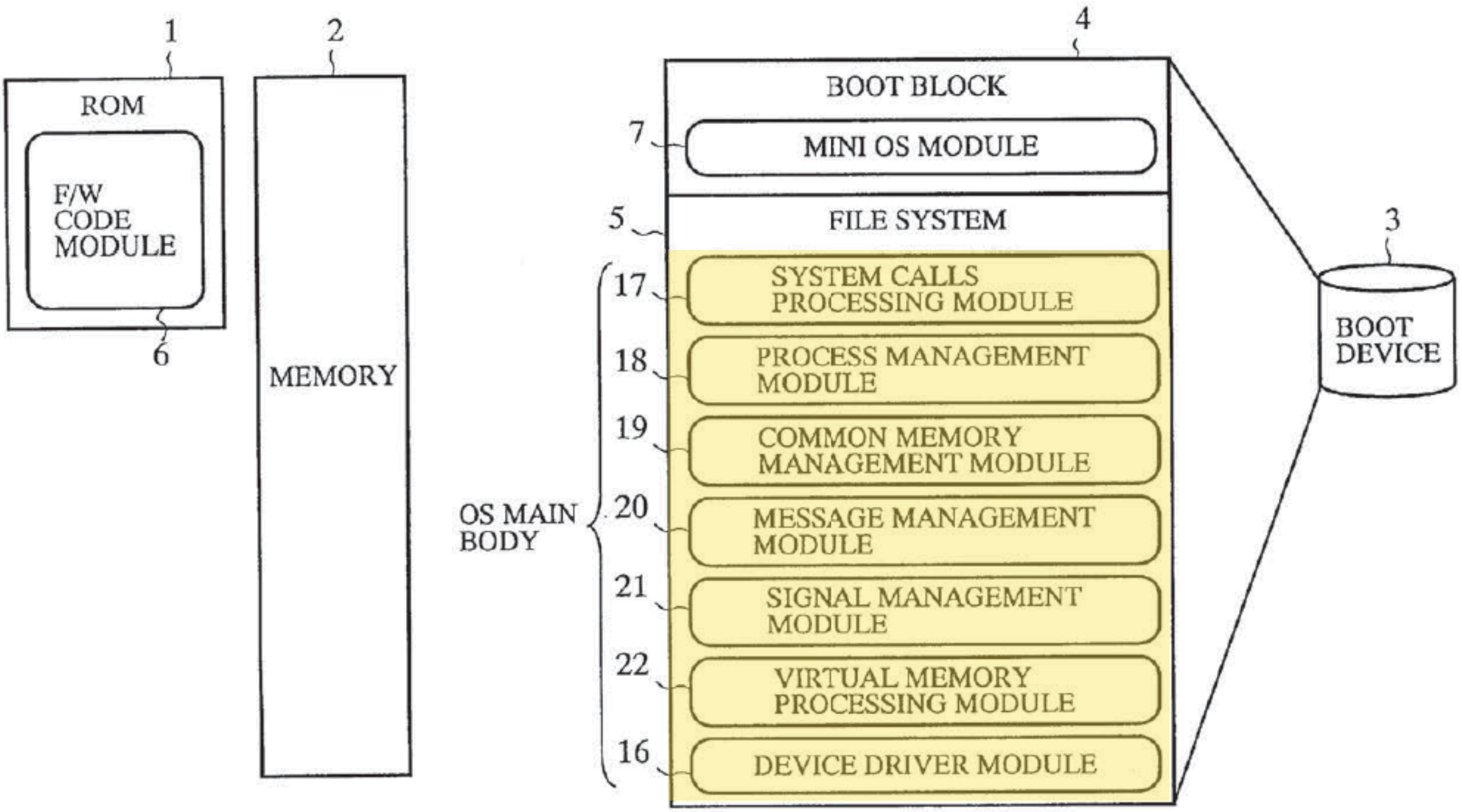
utilizing the decompressed boot data to at least partially boot the computer system; and

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 Patent at Claim 5

SETTSU'S OS MAIN BODY INCLUDES SEVEN OR MORE MODULES ('1739 IPR)



Settsu at Figure 5

SETTSU FAILS TO RENDER OBVIOUS "UPDATING THE BOOT DATA LIST" ('1739 IPR)

78. Instead, a POSITA would have understood that most, if not all, operating system updates would not change the list of OS modules needed to load the first application program in Settsu's system, which is all that is contained in Settsu's "function definition files." For instance, if any of the modules shown in Figure 17 were to be updated (the System Calls Processing Module, Process Management Module, etc.), and if any subset of those modules were needed to load the first application program, this subset would likely not change due to an OS module update. Thus, the function definition file (the claimed "boot data list") would not need to be updated during an operating system update because its contents would not change.

Dr. Back '1739 Declaration (Ex. 2008) at ¶ 78

EXAMINER CONSIDERED SETTSU DURING ORIGINAL PROSECUTION, INCLUDING PASSAGES CITED BY APPLE

EXAMINER'S OFFICE ACTION:

5. Claims 1, 6-7, 9, 12, 14-22, 25-30, 33, 37-48, 51, 55-66, 69, 73-84, 87, 91-102, 105, 109-120, 123 and 127-134 are rejected under pre-AIA 35 U.S.C. 102(e) as being anticipated by Settsu et al (US Patent 6,374,353[1]; hereinafter Settsu).

6. As per claim 1, Settsu disclose a method for providing accelerated loading of an operating system in a computer system, comprising:

maintaining a list of boot data for booting the computer system, wherein at least a portion of boot data is associated with the list of boot data [col. 3, lines 56-59; required modules are listed; col. 16, lines 26-30; col. 16, line 57 -- col. 17, line 20];

'862 File History (Ex. 1002) at 306

EXAMINER SPECIFIED SETTSU DOES NOT TEACH OR SUGGEST "UPDATING THE BOOT DATA LIST"

EXAMINER'S OFFICE ACTION REFERRING TO ALLOWABLE CLAIM 22:

Allowable Subject Matter

52. Claims 2-4 and 22-24 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

22. (Previously Presented) The method of claim 15, further comprising:

updating a list of the boot data.

'862 File History (Ex. 1002) at 288, 411

ZWIEGINCEW SWAPS APPLICATION FILES USING VIRTUAL MEMORY MANAGER

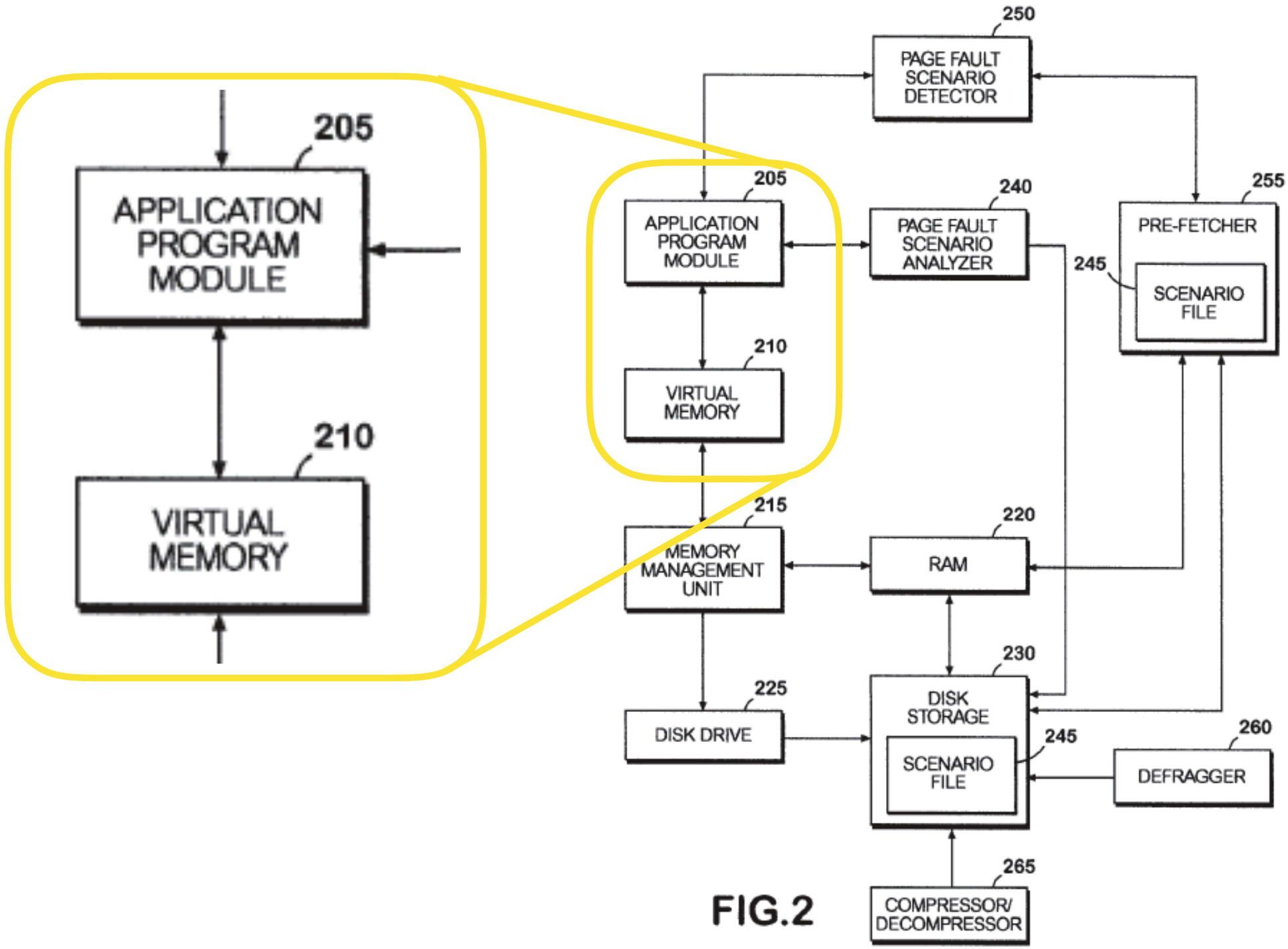


FIG.2

Zwiegincew at Figure 2

ZWIEGINCEW DOES NOT RELATE TO COMPUTER SYSTEM BOOT UP

Zwiegincew does not describe, or relate to, improving boot speed or prefetching of pages expected during the boot process. Rather, a POSITA would have understood that Zwiegincew's teachings relate to managing virtual memory through swapping data pages into RAM from disk memory and swapping data pages out of RAM into disk memory. Because the use of these techniques require the presence of a fully initialized operating system kernel, they can be applied only after the operating system kernel has booted up, and thus cannot relate to the accelerated loading of the operating system kernel itself or to otherwise managing the boot-up process. As such, a POSITA would not have considered Zwiegincew's teachings relevant to "a boot data list" or loading boot data to decrease boot time. Specifically, a POSITA would have understood that Zweigincew's teachings cannot be used to speed up the loading of an operating system for the simple reason that Zwiegncew's techniques *require* the presence of a functioning operating system in order to work.

Dr. Back '1739 Declaration (Ex. 2008) at ¶ 82

ZWIEGINCEW DOES NOT RELATE TO COMPUTER SYSTEM BOOT UP

85. Overall, a POSITA would have understood that Zwiegincew is not concerned with and does not recognize a problem with slow boot speeds due to hard page faults during the boot process. Zwiegincew, indeed, states in the Background of the Invention that "there remains a need for a method and system for improving the performance of an application program module"—as opposed to boot up processes—"by reducing disk access time without burdening the programmer."[57]

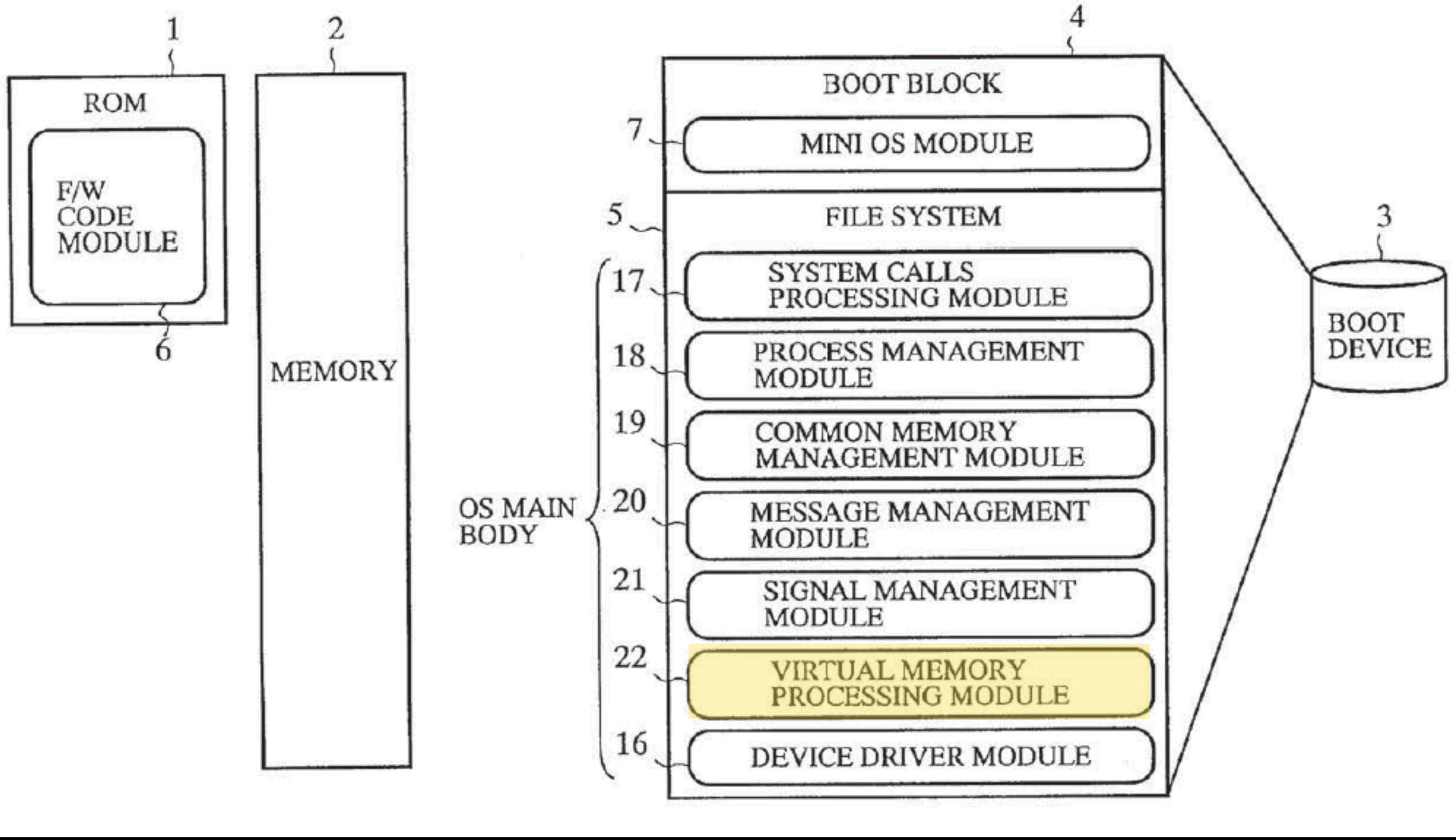
Dr. Back '1739 Declaration (Ex. 2008) at ¶ 85

ZWIEGINCEW DOES NOT RELATE TO COMPUTER SYSTEM BOOT UP

84. Regarding the last passage referenced by Dr. Neuhauser, Zwiegincew's passage at 2:12-15 does not relate to solving slow boot times. This passage begins by explaining "strategic ordering" in which the order of pages likely to be accessed "during typical usage of an application" is determined. Here, Zwiegincew is not discussing acceleration of hard page faults during the boot process of the OS itself. Rather, this approach refers to starting application programs. The "strategic ordering" approach could indeed be useful just after boot, i.e., "the boot [page fault] scenario," but it would not be useful during boot. This passage appears to be the only time Zwiegincew refers to system "boot," or to system start-up in general. Even here, however, the "strategic ordering" referenced in this passage relates to hard page faults caused by application programs. Dr.

Dr. Back '1739 Declaration (Ex. 2008) at ¶ 84

ZWIEGINCEW'S TECHNIQUE COULD NOT BE USED UNTIL SETTSU'S VIRTUAL MEMORY MANAGER HAS BEEN ENABLED ('1739 IPR)



Settsu at Figure 5

DR. NEUHAUSER ADMITS THAT ZWIEGINCEW'S TECHNIQUE REQUIRES AN ENABLED VIRTUAL MEMORY MANAGER

AS REALTIME ARGUES IN ITS RESPONSES, ZWIEGINCEW'S TECHNIQUE REQUIRES AN ENABLED VIRTUAL MEMORY MANAGER:

Q. Is that what Zwiegincew teaches expressly, doing that it way?

A. I don't think it says one way or the other. It just says that you have -- basically, Zwiegincew assumes that the virtual memory is available to it when it needs it. So in that sense, it's enabled when it -- it certainly has to be enabled by the time -- by the time you leave the loading process in Zwiegincew or preloading, if you will.

Dr. Neuhauser Testimony (Ex. 2024) at 103:6-12; see also '1737 Patent Owner Response at 15 & 38, '1738 Patent Owner Response at 14 & 36-37, '1739 Patent Owner Response at 31-32 & 35-36

DR. NEUHAUSER ADMITS THAT SETTSU'S VIRTUAL MEMORY MANAGER IS NOT ENABLED UNTIL TRANSFERRING INTO MEMORY 2

AS REALTIME ARGUES IN ITS RESPONSE, SETTSU'S VIRTUAL MEMORY MANAGER MODULE IS ENABLED AFTER IT HAS BEEN TRANSFERRED TO MEMORY 2:

Q. Before the OS main body module has moved from boot device 3 into memory 2, are the modules within the OS main body module enabled?

A. What do you mean by "enabled" here?

Q. Are they booted, loading and running?

MR. HUGUENIN-LOVE: Object to form.

THE WITNESS: I don't believe so.

Dr. Neuhauser Testimony (Ex. 2024) at 105:17-23; see also see also '1739 Patent Owner Response at 7, 35-36

NO MOTIVATION EXISTS TO COMBINE SETTSU AND ZWIEGINCEW

87. With respect to the motivation to combine Settsu and Zwiegincew, a POSITA would have understood that hard page fault handling does not apply to the process of loading the operating system itself. Rather, page fault handling is implemented by the operating system's virtual memory processing module or manager, which is necessarily available only after the boot up process. Zwiegincew confirms this fact, stating that "[i]n the context of a paging memory system, a "page" is defined as a fixed-size block of bytes whose physical address can be changed via the MMU, working in conjunction with a Virtual Memory Manager."[60] This explains why "boot" is not discussed anywhere in Zwiegincew, except in one passing reference in the background section, which however indicates that application programs may cause page faults shortly after boot.

Dr. Back '1739 Declaration (Ex. 2008) at ¶ 87

NO MOTIVATION EXISTS TO COMBINE SETTSU AND ZWIEGINCEW

88. Moreover, Zwiegincew's page swapping method, which requires an initialized virtual memory manager, is antithetical to Settsu's teachings. Settsu teaches speeding up the boot process by dividing the OS into a mini-OS module with some basic functionality, which then bootstraps the loading of the OS main module. In Figures 3, 5, 12, and 17 of Settsu, this virtual memory processing module 22 is part of the OS main body module. For example, Settsu states with

89. Based on this description, a POSITA would have understood that the virtual memory manager, which is need to implement Zwiegincew's page swapping technique, is *not available to use* to speed up Settsu's boot process since it is part of the main OS. For this simple reason, a POSITA could not have combined Zwiegincew's updating of the scenario file with Settsu's purported boot list.

Dr. Back '1739 Declaration (Ex. 2008) at ¶¶ 88-89

EXAMINER CONSIDERED ZWIEGINCEW AND SETTSU DURING ORIGINAL PROSECUTION

References Cited

6,317,714	B1	11/2001	Del Castillo et al.
6,317,818	B1	11/2001	Zwiegincew et al.
6,330,622	B1	12/2001	Schaefer
6,333,745	B1	12/2001	Shimomura et al.
6,336,153	B1	1/2002	Izumida et al.
6,345,307	B1	2/2002	Booth
6,356,589	B1	3/2002	Gebler et al.
6,356,937	B1	3/2002	Montville et al.
6,374,353	B1	4/2002	Settsu et al.
6,388,584	B1	5/2002	Dorward et al.
6,392,567	B2	5/2002	Satoh
6,404,931	B1	6/2002	Chen et al.
6,421,387	B1	7/2002	Rhee

'862 Patent at page 4

IPR2016-01739 ISSUE

SETTSU AND ZWIEGINCEW DO NOT RENDER OBVIOUS “UPDATING THE BOOT DATA LIST IN RESPONSE TO THE UTILIZING” STEP

CLAIM 112'S "UPDATING THE BOOT DATA LIST IN RESPONSE TO THE UTILIZING" ('1739 IPR)

5. A method for booting a computer system, the method comprising:

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;

accessing the loaded compressed boot data;

decompressing the accessed compressed boot data;

utilizing the decompressed boot data to at least partially boot the computer system; and

updating the boot data list,

112. The method of claim **5**, wherein the updating comprises:

updating the boot data list in response to the utilizing.

'862 Patent at claims 5 & 112

ZWIEGINCEW DOES NOT SUGGEST UPDATING A BOOT DATA LIST IN RESPONSE TO UTILIZING DECOMPRESSED BOOT DATA

104. Specifically, a POSITA would not have considered Zwiegincew's teachings toward scenario files or to fetching pages before hard page faults relevant to claim 5's "utilizing" step nor to a "boot data list." Indeed, Dr. Neuhauser does not refer to Zwieginciew's teachings when analyzing claim 5's "utilizing" step (element 5.5).[69]

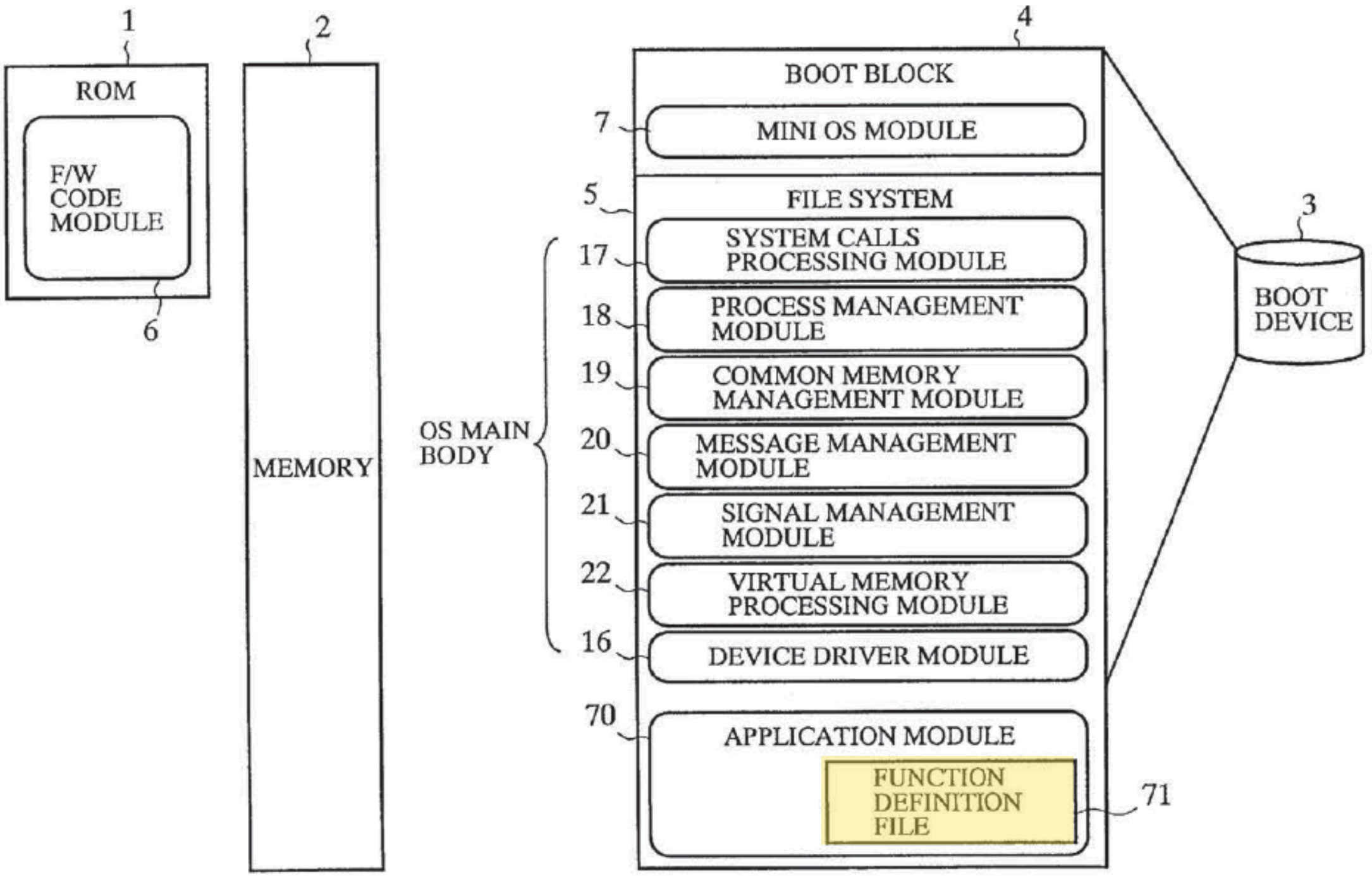
Dr. Back '1739 Declaration (Ex. 2008) at ¶ 104

ZWIEGINCEW DOES NOT SUGGEST UPDATING A BOOT DATA LIST IN RESPONSE TO UTILIZING DECOMPRESSED BOOT DATA

106. Likewise, a POSITA would not have considered refining Zwiegincew's scenario file meets "updating the boot data list in response to" the "utilizing" step of claim 5. Rather, a POSITA would have understood that the claimed step of "utilizing…boot data to at least partially boot the computer system" is unrelated to Zwiegincew's scenario file. Dr. Neuhauser refers to Zwiegincew's description of pattern-based subsequent refinement of scenario files as a suggestion to add or remove data based upon use.[70] However, a POSITA would not have considered Zwiegincew's pattern-based refinement relevant to claim 5's step of utilizing decompressed boot data for system boot-up nor to updating a boot data list in response to the utiziling step, as required by claim 112.

Dr. Back '1739 Declaration (Ex. 2008) at ¶ 106

SETTSU'S FUNCTION DEFINITION FILE IS NOT RELATED TO APPLICATION FILE SWAPPING



Settsu at Figure 17

NO MOTIVATION EXISTS TO COMBINE SETTSU AND ZWIEGINCEW TO RENDER OBVIOUS CLAIM 112

107. Dr. Neuhauser's analysis regarding claim 112 also does not explain why or how a POSITA would have used Zwiegincew's teachings regarding page swapping and scenario files to modify Settsu's system to update the functional definition file 71 (the alleged "boot data list"). This is likely because Settsu's function definition file (the purported "boot data list" in Dr. Neuhauser's hypothetical construction) would not be updated in response to the utilizing.

Dr. Back '1739 Declaration (Ex. 2008) at ¶ 107

IPR2016-01739 ISSUE

“NON-ACCESSED BOOT DATA” CONSTRUCTION

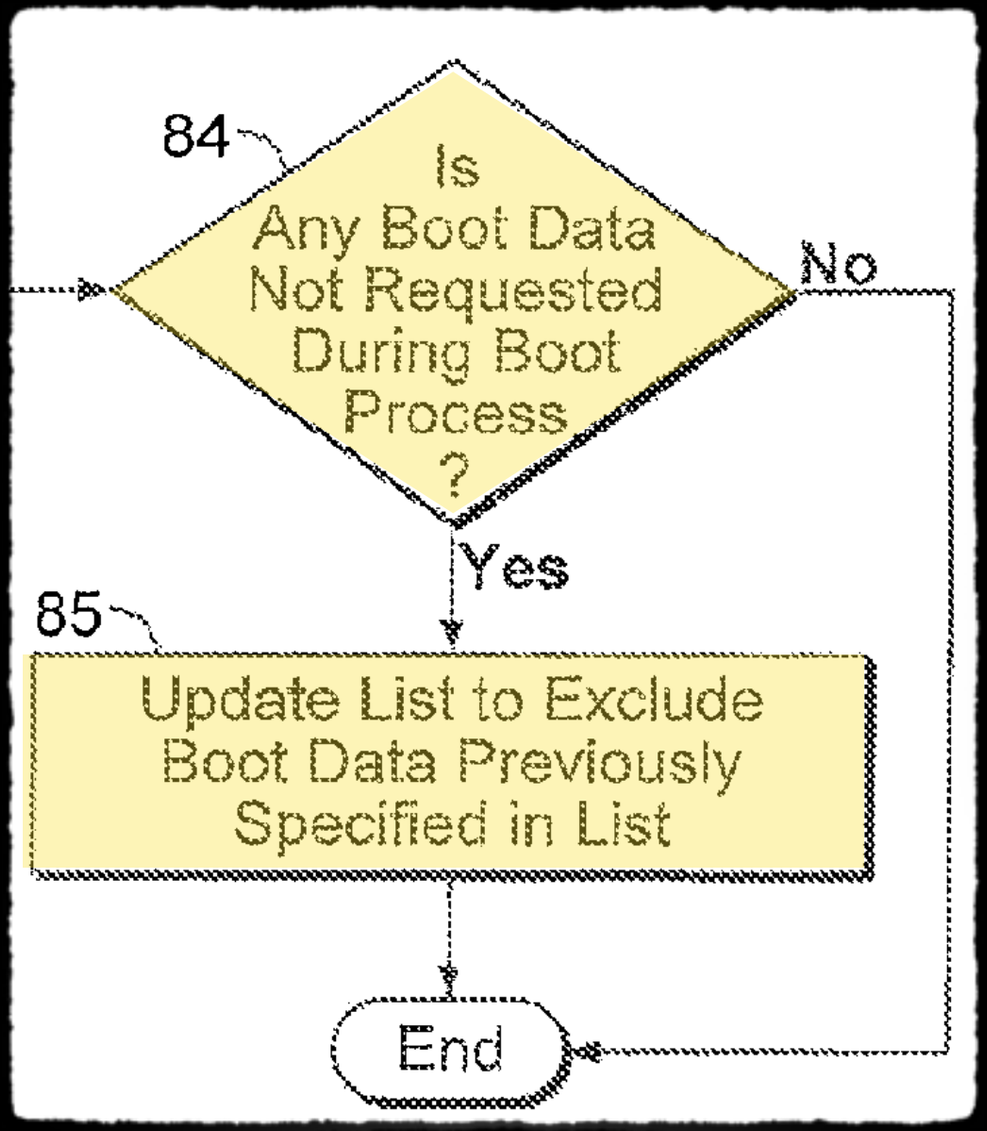
"NON-ACCESSED BOOT DATA" CONSTRUCTION

98. The method of claim **5**, wherein the updating comprises:

disassociating non-accessed boot data from the boot data list.

'862 Patent at claim 98

‘862 SPECIFICATION DISCLOSES “NON-ACCESSED BOOT DATA” IS IDENTIFIED IN BOOT DATA LIST BUT NOT ACCESSED DURING BOOT



‘862 Patent at Figure 7B

Further, during the boot process, if no request is made by the host computer for a data block that was pre-loaded into the local memory of the data storage controller (affirmative result in step **84**), then the boot data list will be updated by removing the non-requested data block from the list (step **85**). Thereafter, upon the next boot sequence, the data storage controller will not pre-load that data into local memory.

‘862 Patent at 22:5-11

POSITA WOULD UNDERSTAND "NON-ACCESSED BOOT DATA" IS IDENTIFIED IN BOOT DATA LIST BUT NOT ACCESSED DURING BOOT

62. Based on the '862 Patent, a POSITA would have understood that the recited "non-accessed boot data list" refers to boot data identified in the boot data list that was not requested during system boot-up." The '862 specification discloses that "non-accessed boot data" is boot data that has been retrieved and recorded in the boot data list during a previous system boot-up, but was not requested during a subsequent system boot-up. For example, the specification

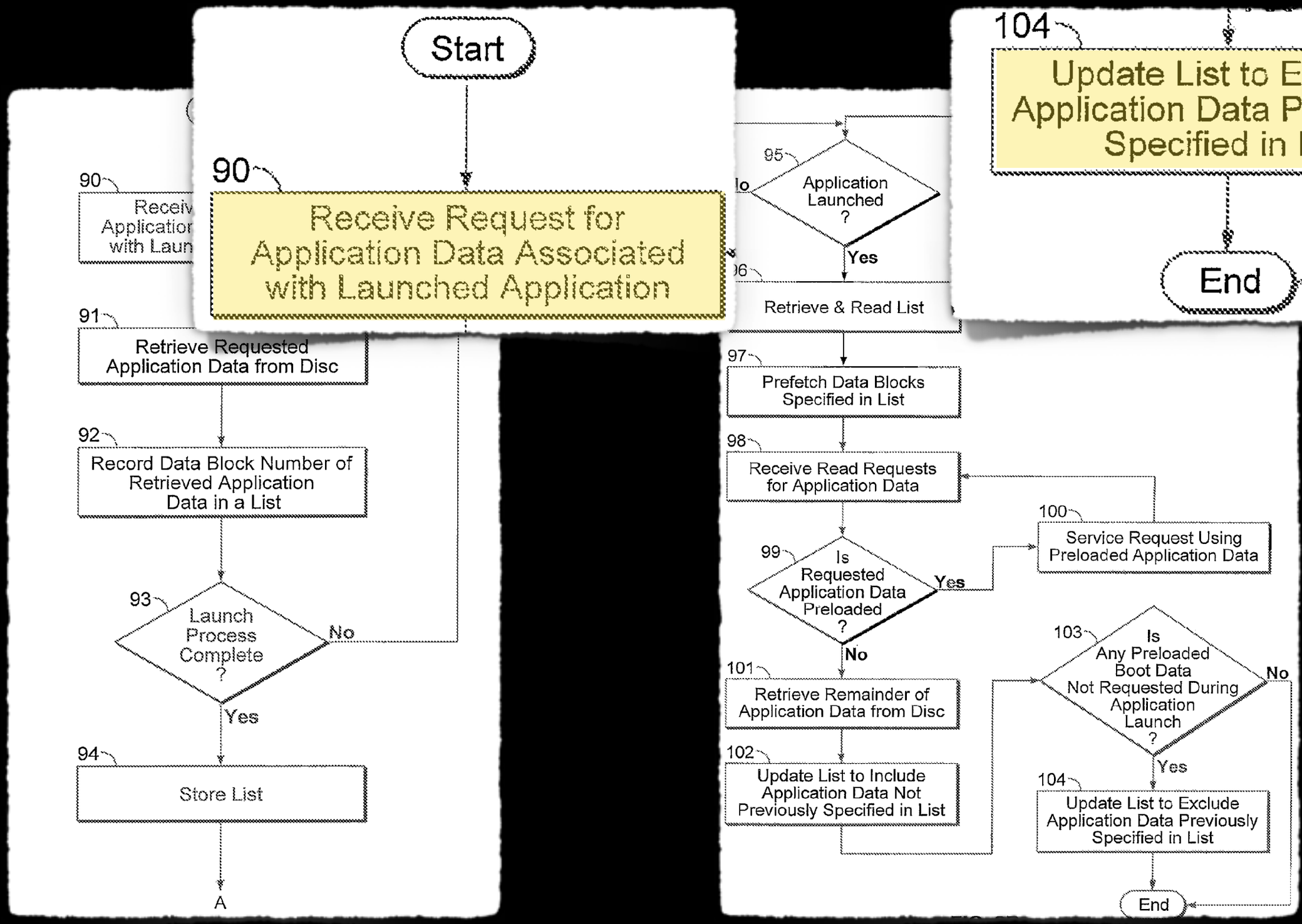
Dr. Back '1739 Declaration (Ex. 2008) at ¶ 62

"NON-ACCESSED BOOT DATA" MEANS "BOOT DATA IDENTIFIED IN THE BOOT DATA LIST THAT WAS NOT REQUESTED DURING SYSTEM BOOT-UP"

66. Therefore, in my opinion, the broadest reasonable interpretation of "non-accessed boot data" in light of the specification is "boot data identified in the boot data list that was not requested during system boot-up."

Dr. Back '1739 Declaration (Ex. 2008) at ¶ 66

‘862 EMBODIMENT TOWARD QUICK LAUNCH OF APPLICATION PROGRAMS IS USED AFTER SYSTEM BOOT-UP



‘862 Patent at Figures 8A-8B

APPLE MISCONSTRUES PROSECUTION HISTORY WHEN CONSTRUING "NON-ACCESSED BOOT DATA"

Further, the portion of the '862 patent cited in Realtime's POR and the prosecution history as supporting "non-accessed boot data" includes references to a "non-requested data block" "[d]uring the *application launch process*." APPLE-1001, 22:12-23:26; APPLE-1002 (Part 1), 156-157, 160-162. Thus, Realtime's own

Apple's '1739 Reply at 6

APPLE RELIES ON PROSECUTION HISTORY WHEN CONSTRUING "NON-ACCESSED BOOT DATA"

APPLE'S PROSECUTION EVIDENCE RELATES TO LOADING "APPLICATION DATA" (NOT BOOT DATA) TO UPDATE AN "APPLICATION DATA LIST" (NOT A BOOT DATA LIST):

As a yet further non-limiting example, Applicant respectfully directs the Examiner to ¶ [0115] of the Specification that provides, with emphasis added:

> [0115] Further, during the launch process, if no request is made by the host computer for a data block that was pre-loaded into the local memory of the data storage controller (affirmative result in step 103), then the *application data list will be updated by removing the non-requested data block from the list* (step 104). Thereafter, upon the next launch sequence for the given application, the data storage controller will not pre-load that data into local memory.

'862 File History (Ex. 1002) at 157, 162

IPR2016-01739 ISSUE

SETTSU AND ZWIEGINCEW DO NOT RENDER OBVIOUS "DISASSOCIATING NON-ACCESSED BOOT DATA FROM THE BOOT DATA LIST"

ZWIEGINCEW DOES NOT SUGGEST "DISASSOCIATING NON-ACCESSED BOOT DATA" ELEMENT

94. First, a POSITA would not have considered Zwiegincew's page swapping teachings relevant to the claimed boot-up method or to changes to Zwiegincew's scenario file. Dr. Neuhauser references only one passage of Zwiegincew (6:20-25) regarding page swapping. However, as Dr. Neuhauser admitted during his deposition, that referenced passage does not address Zwiegincew's scenario file and does not discuss boot data.[65] Instead, as Dr. Neuhauser confirmed, Zwiegincew's passage (6:20-25) generally discusses memory management performed by a virtual memory manager.[66] The passage

Dr. Back '1739 Declaration (Ex. 2008) at ¶ 94

ZWIEGINCEW DOES NOT SUGGEST "DISASSOCIATING NON-ACCESSED BOOT DATA" ELEMENT

97. A POSITA would have understood that the claimed "non-accessed boot data" and "accessed boot data" are concepts unrelated to the scenario file taught by Zwiegincew because scenario files cannot be used during boot-up. Moreover, Zwiegincew's page files in the scenario file are unrelated to boot-up. Therefore, any "updating" taught by Zwiegincew does not meet the claimed "non-accessed boot data," which requires "boot data" not be requested during system boot-up.

Dr. Back '1739 Declaration (Ex. 2008) at ¶ 97

APPLE RELIES ON ZWIEGINCEW'S 6:20-25 FOR "DISASSOCIATING NON-ACCESSED BOOT DATA" ELEMENT

Zwiegincew further describes that "the requested pages are swapped with less used pages in the RAM 220" such that "the least recently used pages are stored on the disk storage 230." Zwiegincew, 6:20-25. From this and related description,

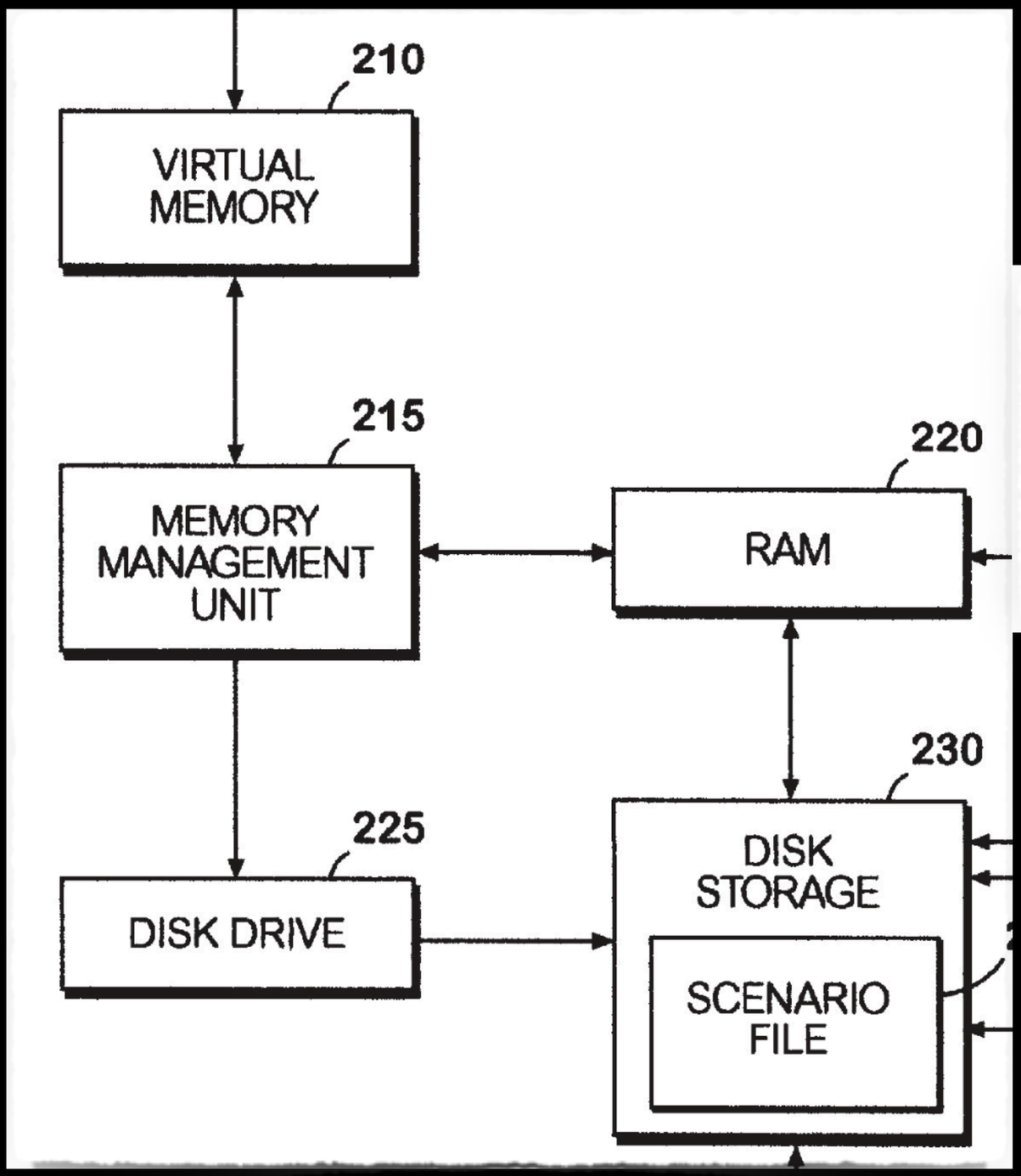
'1739 Petition at 58-59

DR. NEUHAUSER RELIES ON ZWIEGINCEW'S 6:20-25 FOR "DISASSOCIATING NON-ACCESSED BOOT DATA" ELEMENT

Zwiegincew further describes that "the requested pages are swapped with less used pages in the RAM 220" such that "the least recently used pages are stored on the disk storage 230," with corresponding changes to implicated scenario files [Zwiegincew, 6:20-25]. From this and related description, one of ordinary skill would have been motivated to make room in memory 2 for frequently requested boot data by removing least recently used data from memory 2. Further, from

Dr. Neuhauser '1739 Declaration (Ex. 1003) at ¶ 204

ZWIEGINCEW'S 6:20-25 IS UNRELATED TO DISASSOCIATING NON-ACCESSED BOOT DATA



Zwiegincew at Figure 2

transfer the requested pages into the RAM **220**. Typically, the requested pages are swapped with less used pages in the RAM **220**. Accordingly, the least recently used pages are stored on the disk storage **230** and the VMM updates its records to reflect the new pages in the RAM **220**. Swapping is a memory management technique that is well known in the art. Those skilled in the art will appreciate that above

Zwiegincew at 6:20-25

DR. NEUHAUSER ADMITS ZWIEGINCEW'S 6:20-25 IS UNRELATED TO "DISASSOCIATING NON-ACCESSED BOOT DATA" ELEMENT

Q My point is -- is a simple one. Lines 20 through 25 that you --

A Uh-huh.

Q -- cite in Paragraph 204, those lines don't discuss attributes of Zwiegincew's scenario file?

A Yes, that's correct.

Q And these lines don't discuss boot data?

MR. BITTNER: I'll object to the form.

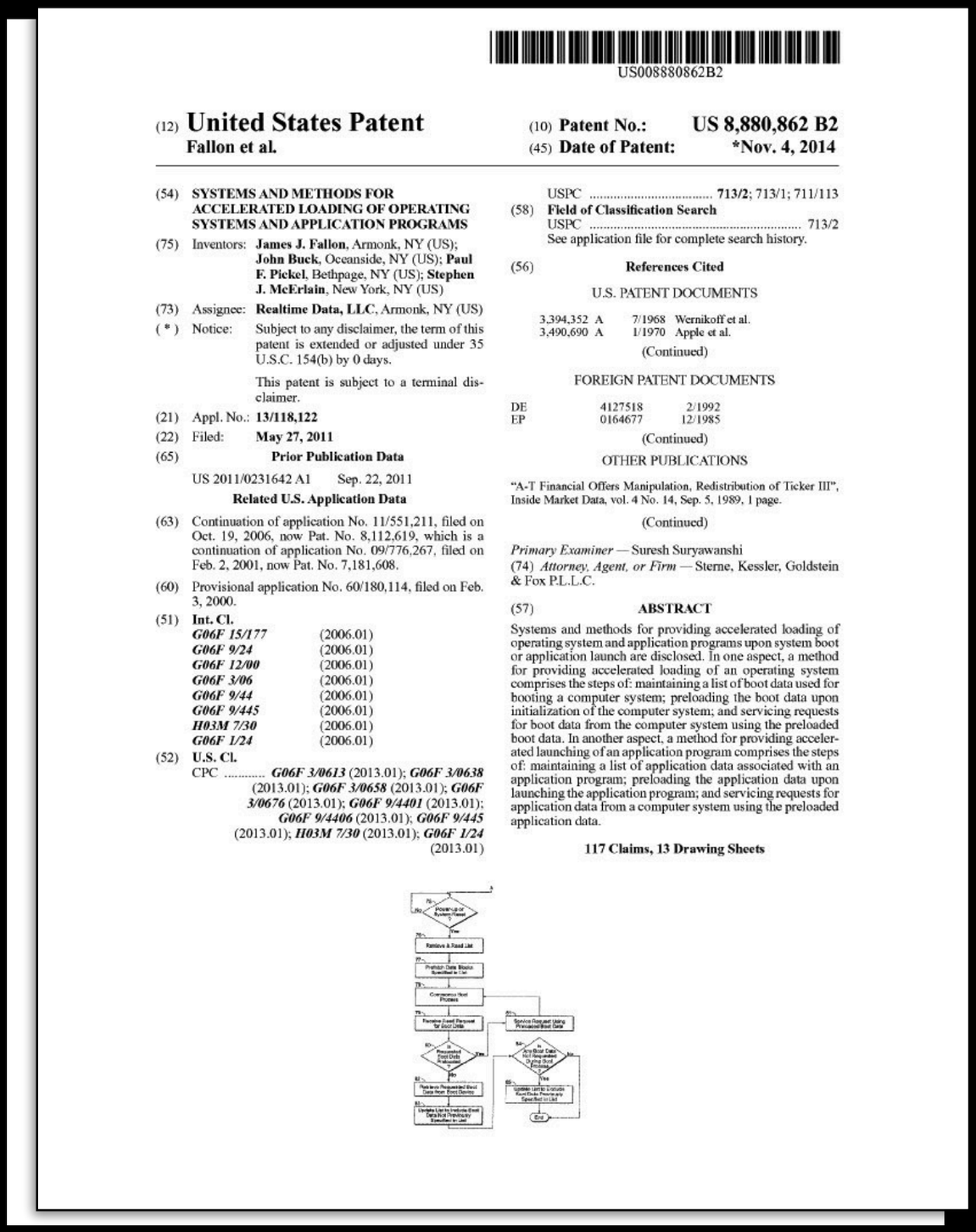
A I don't think they use the word "boot data." They're not discussing boot data here.

Dr. Neuhauser Transcript (Ex. 2011) at 179:9-18

IPR2016-01737, -01738 ISSUE

SUKEGAWA AND ZWIEGINCEW DO NOT RENDER OBVIOUS A “BOOT DATA LIST”

GROUNDS 1-5 IN EACH ‘1737 IPR AND ‘1738 IPR FAIL TO RENDER OBVIOUS THE ‘862 CLAIMS



US008880862B2

(12) **United States Patent**
Fallon et al.

(10) **Patent No.: US 8,880,862 B2**
(45) **Date of Patent: *Nov. 4, 2014**

(54) **SYSTEMS AND METHODS FOR ACCELERATED LOADING OF OPERATING SYSTEMS AND APPLICATION PROGRAMS**

(75) Inventors: **James J. Fallon**, Armonk, NY (US); **John Buck**, Oceanside, NY (US); **Paul F. Pickel**, Bethpage, NY (US); **Stephen J. McErlain**, New York, NY (US)

(73) Assignee: **Realtime Data, LLC**, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **13/118,122**

(22) Filed: **May 27, 2011**

(65) **Prior Publication Data**

US 2011/0231642 A1 Sep. 22, 2011

Related U.S. Application Data

(63) Continuation of application No. 11/551,211, filed on Oct. 19, 2006, now Pat. No. 8,112,619, which is a continuation of application No. 09/776,267, filed on Feb. 2, 2001, now Pat. No. 7,181,608.

(60) Provisional application No. 60/180,114, filed on Feb. 3, 2000.

(51) **Int. Cl.**

G06F 15/177	(2006.01)
G06F 9/24	(2006.01)
G06F 12/00	(2006.01)
G06F 3/06	(2006.01)
G06F 9/44	(2006.01)
G06F 9/445	(2006.01)
H03M 7/30	(2006.01)
G06F 1/24	(2006.01)

(52) **U.S. Cl.**

CPC ***G06F 3/0613*** (2013.01); ***G06F 3/0638*** (2013.01); ***G06F 3/0658*** (2013.01); ***G06F 3/0676*** (2013.01); ***G06F 9/4401*** (2013.01); ***G06F 9/4406*** (2013.01); ***G06F 9/445*** (2013.01); ***H03M 7/30*** (2013.01); ***G06F 1/24*** (2013.01)

USPC **713/2**; 713/1; 711/113

(58) **Field of Classification Search**

USPC 713/2

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,394,352 A 7/1968 Wernikoff et al.
3,490,690 A 1/1970 Apple et al.

(Continued)

FOREIGN PATENT DOCUMENTS

DE 4127518 2/1992
EP 0164677 12/1985

(Continued)

OTHER PUBLICATIONS

"A-T Financial Offers Manipulation, Redistribution of Ticker III", Inside Market Data, vol. 4 No. 14, Sep. 5, 1989, 1 page.

(Continued)

Primary Examiner — Suresh Suryawanshi
(74) *Attorney, Agent, or Firm* — Sterne, Kessler, Goldstein & Fox P.L.L.C.

(57) **ABSTRACT**

Systems and methods for providing accelerated loading of operating system and application programs upon system boot or application launch are disclosed. In one aspect, a method for providing accelerated loading of an operating system comprises the steps of: maintaining a list of boot data used for booting a computer system; preloading the boot data upon initialization of the computer system; and servicing requests for boot data from the computer system using the preloaded boot data. In another aspect, a method for providing accelerated launching of an application program comprises the steps of: maintaining a list of application data associated with an application program; preloading the application data upon launching the application program; and servicing requests for application data from a computer system using the preloaded application data.

117 Claims, 13 Drawing Sheets

Sukegawa fails to disclose or render obvious claim elements:

- *“loading [or accessing] boot data … that is associated with a boot data list” (cls. 1, 11, 6, 8, 13, 14)*
- *“disassociating non-accessed boot data from the boot data list" (cls. 20, 96, 100, 102, 104, 106)*
- *“the compressed boot data comprises: a program code associated with … and an application program” (cls. 17, 29, 53, 65, 77, 89)*
- *“utilizing the stored additional portion of the operating system to at least further partially boot the computer system” (cl. 9)*

SUKEGAWA'S STORAGE OF A SINGLE BOOT DATA FILE IS NOT A "BOOT DATA LIST"

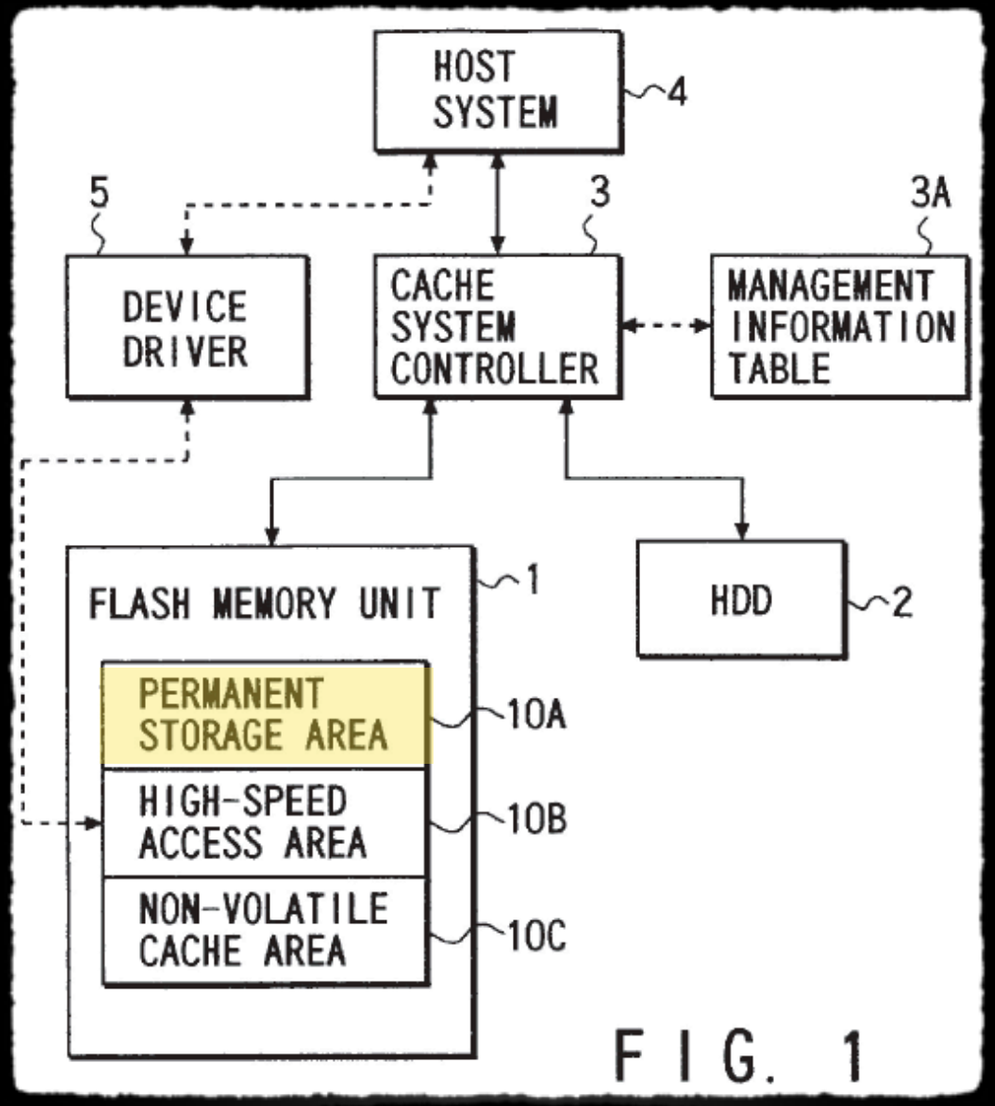


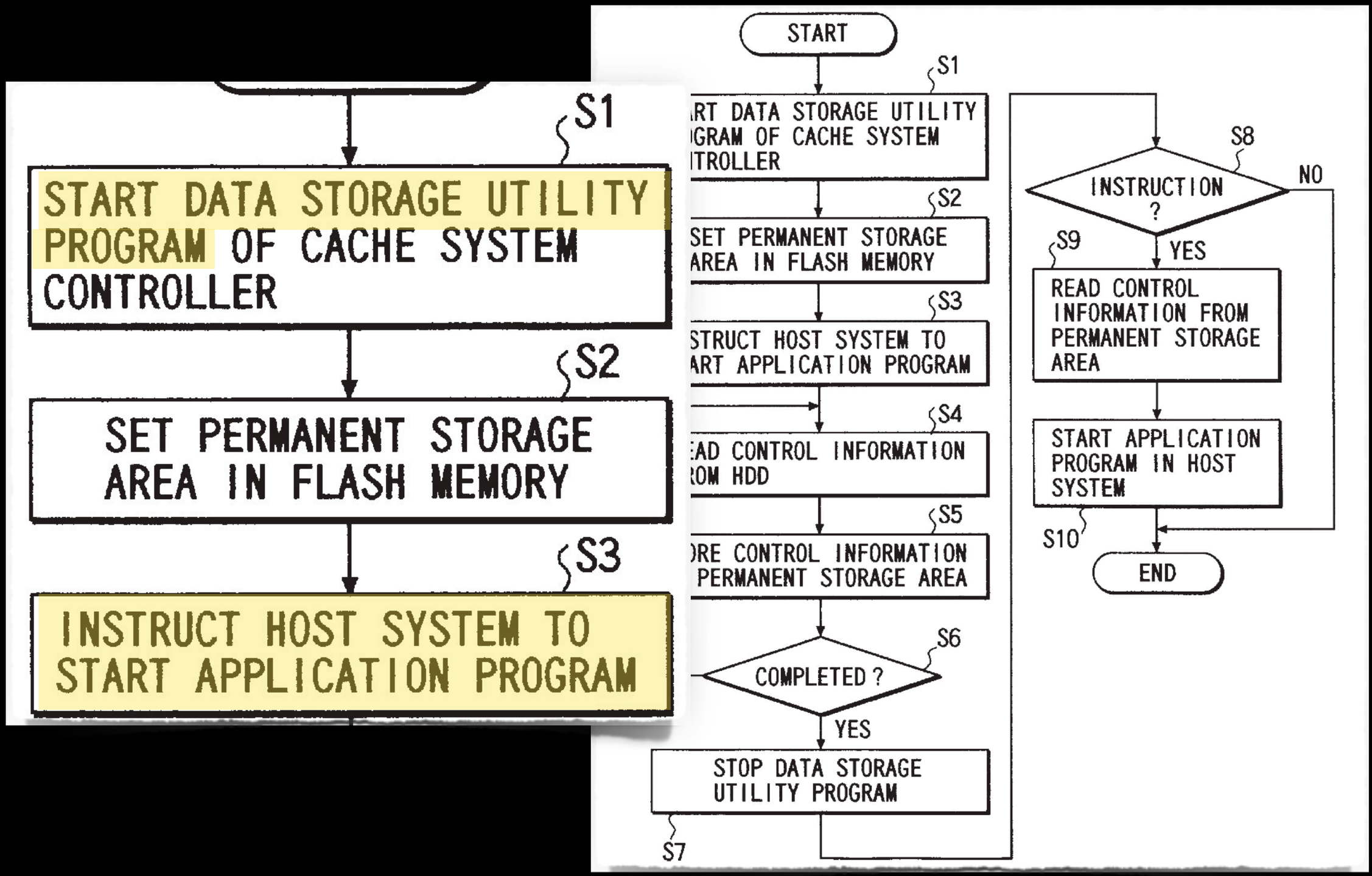
FIG. 1

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 **10A** 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 **10A** 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.

Like the AP control information, the OS control information may be stored as one file in the permanent storage area **10A**. Thereby, the user can refer to, or delete, the OS control information on an as-needed basis. For example, at the time

Sukegawa at 6:45-62

USER INTERFACE MUST BE USED TO ACCESS SUKEGAWA'S APPLICATION PROGRAM INFORMATION FILES STORED IN FLASH



Sukegawa at Fig. 3

USER INTERFACE MUST BE USED TO ACCESS SUKEGAWA'S APPLICATION PROGRAM INFORMATION FILES STORED IN FLASH

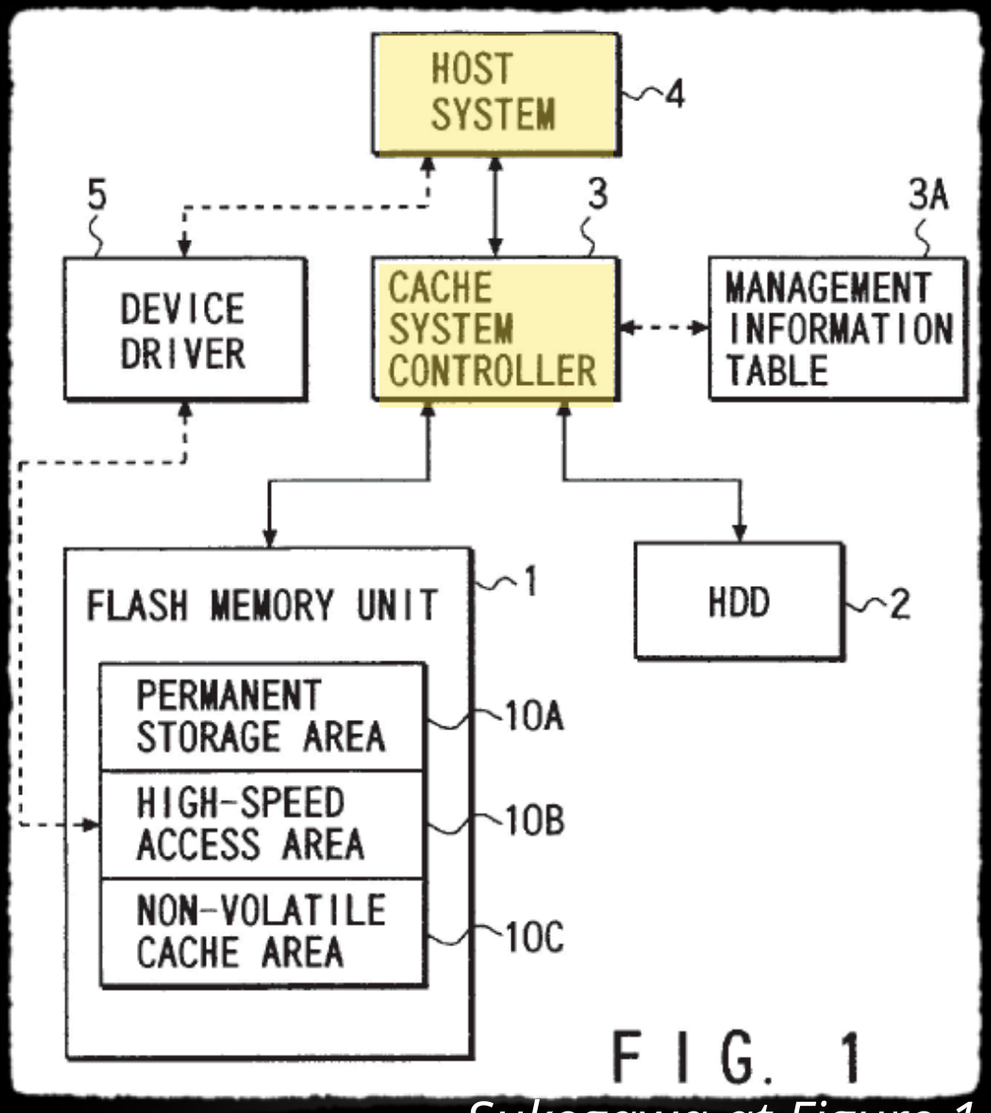


FIG. 1

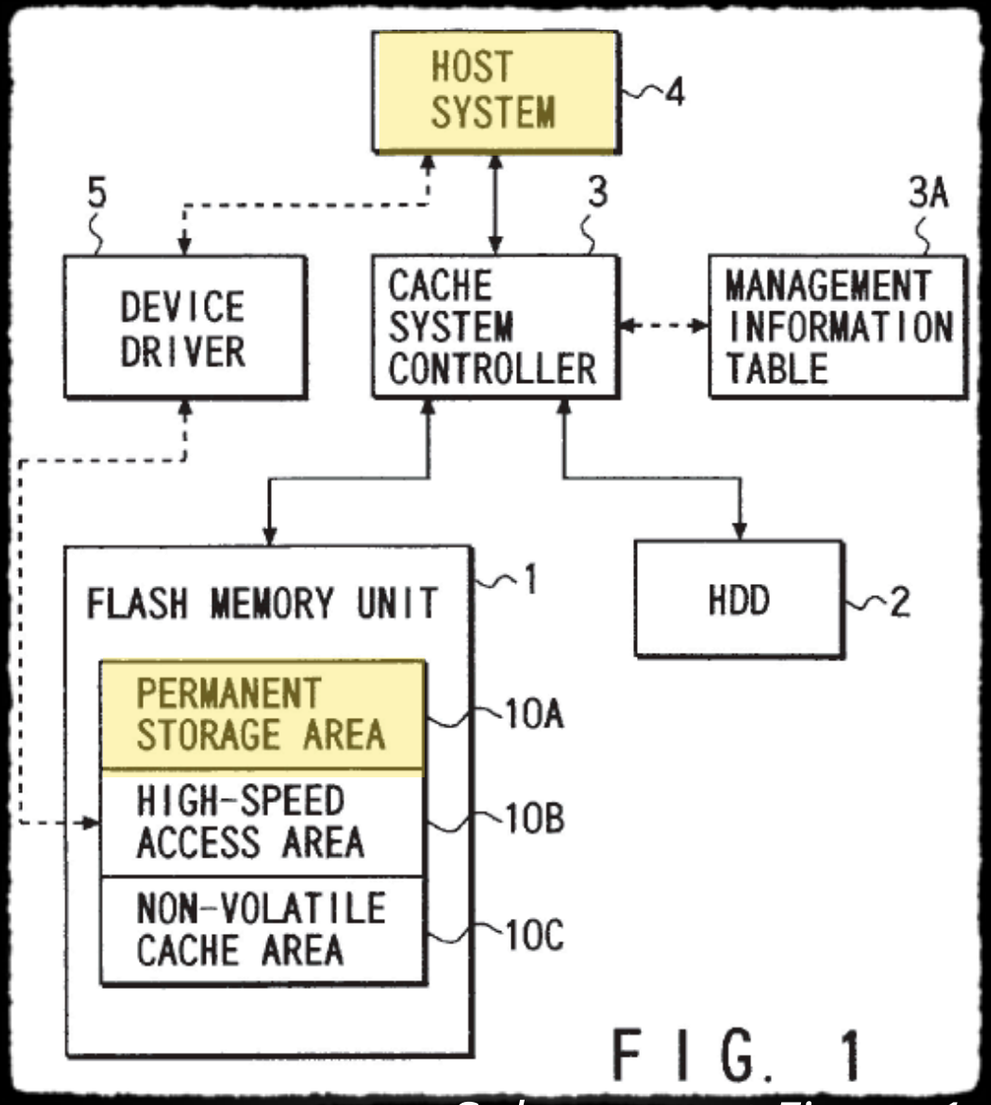
Sukegawa at Figure 1

Then, the user instructs the host system **4** to start the AP (step S**3**). 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**.

If the user instructs the start of the same AP via the user interface, the host system **4** issues the read command, as described above, to read from the HDD **2** the control information necessary for starting the AP ("YES" in step S**8**). Upon receiving the read command, the controller **3**

Sukegawa at 5:24-28, 5:54-58

DR. NEUHAUSER ADMITS SUKEGAWA DISCLOSES ACCESSING APPLICATION PROGRAM INFORMATION VIA USER INTERFACE



Sukegawa at Figure 1

Q So Sukegawa discloses that once the user instructs the host system to -- that it wants to retrieve an application program, that's when the host system sends out a read command, correct?

A That's correct.

In the first embodiment of Sukegawa, Sukegawa discloses that the application program being retrieved from storage area 10A is being retrieved due to an instruction from the user using the user interface, correct?

A I think that's correct. It's kind of roundabout, but I generally agree with it.

Dr. Neuhauser Transcript (Ex. 2011) at 209:20-210:1, 211:11-17

SUKEGAWA'S APPLICATION PROGRAM CONTROL INFORMATION IS NOT "BOOT DATA"

table 3A references information regarding AP control information files. Because Sukegawa discloses that "the user instructs the start of the same AP [stored in flash] via the user interface,"[57] a POSITA would have understood that Sukegawa's user interface is available to the user only after the system has booted-up. Accordingly, a POSITA would not have considered AP control information files stored in flash memory 1 to be "boot data." Thus, a POSITA would not have understood that table 3's reference to information regarding AP control information in Sukegawa meets the claimed "boot data list."

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 82

ZWIEGINCEW TEACHES USING A SCENARIO FILE TO SWAP FILES IN AND OUT OF VIRTUAL MEMORY

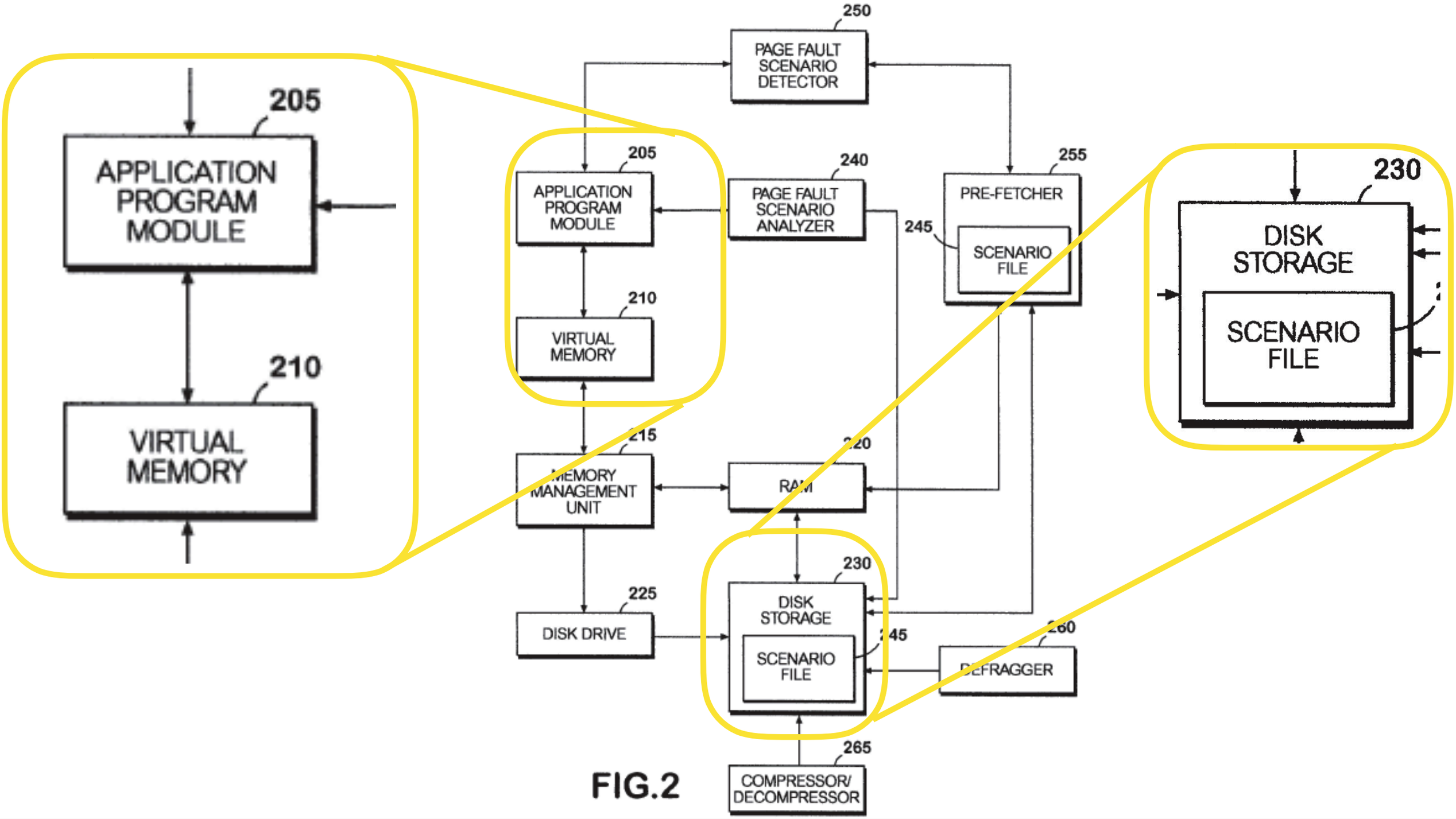


FIG.2

Zwiegincew at Figure 2

ZWIEGINCEW DOES NOT SUGGEST A "BOOT DATA LIST"

Zwiegincew's scenario file to be a "boot data list." A POSITA would understand that such scenario files are employed to address a "scenario" in which a virtual memory manager services a hard page fault interrupt when the CPU detects an access to an unmapped virtual address. The interrupt requires swapping in data for the page into physical memory (RAM) and updating the address of that data in the memory manager unit (MMU).[65] Overall, a POSITA would have understood that a scenario file, such as those disclosed in Zwiegincew, cannot include the operating system itself.

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 90

ZWIEGINCEW DOES NOT SUGGEST A "BOOT DATA LIST"

pages expected during the boot process. Rather, a POSITA would have understood that Zwiegincew's teachings relate to managing virtual memory through swapping data pages into RAM from disk memory and swapping data pages out of RAM into disk memory. Because the use of these techniques requires the presence of a fully initialized operating system kernel, they can be applied only after the operating system kernel has booted up, and thus cannot relate to the accelerated loading of the operating system kernel itself or to otherwise managing the boot-up process. As such, a POSITA would not have considered Zwiegincew's teachings relevant to "a boot data list" or loading boot data to decrease boot time. Specifically, a POSITA would have understood that Zwiegincew's teachings cannot be used to speed up the loading of an operating system for the simple reason that Zwiegincew's techniques *require* the presence of a functioning operating system in order to work.

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 86

SUKEGAWA SPECIFIES MANAGEMENT INFORMATION TABLE 3A ONLY MANAGES FLASH AREAS 10A AND 10C

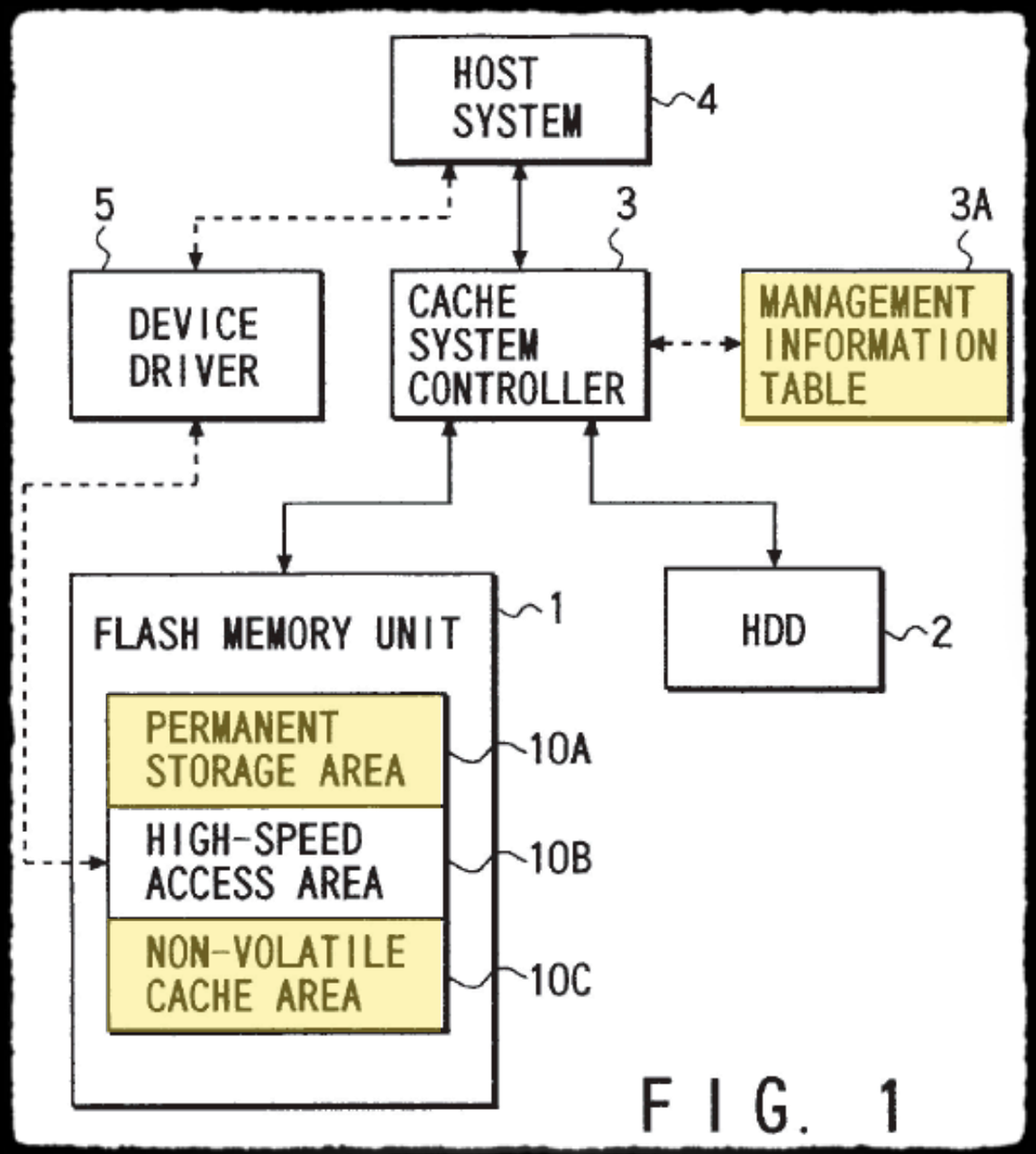


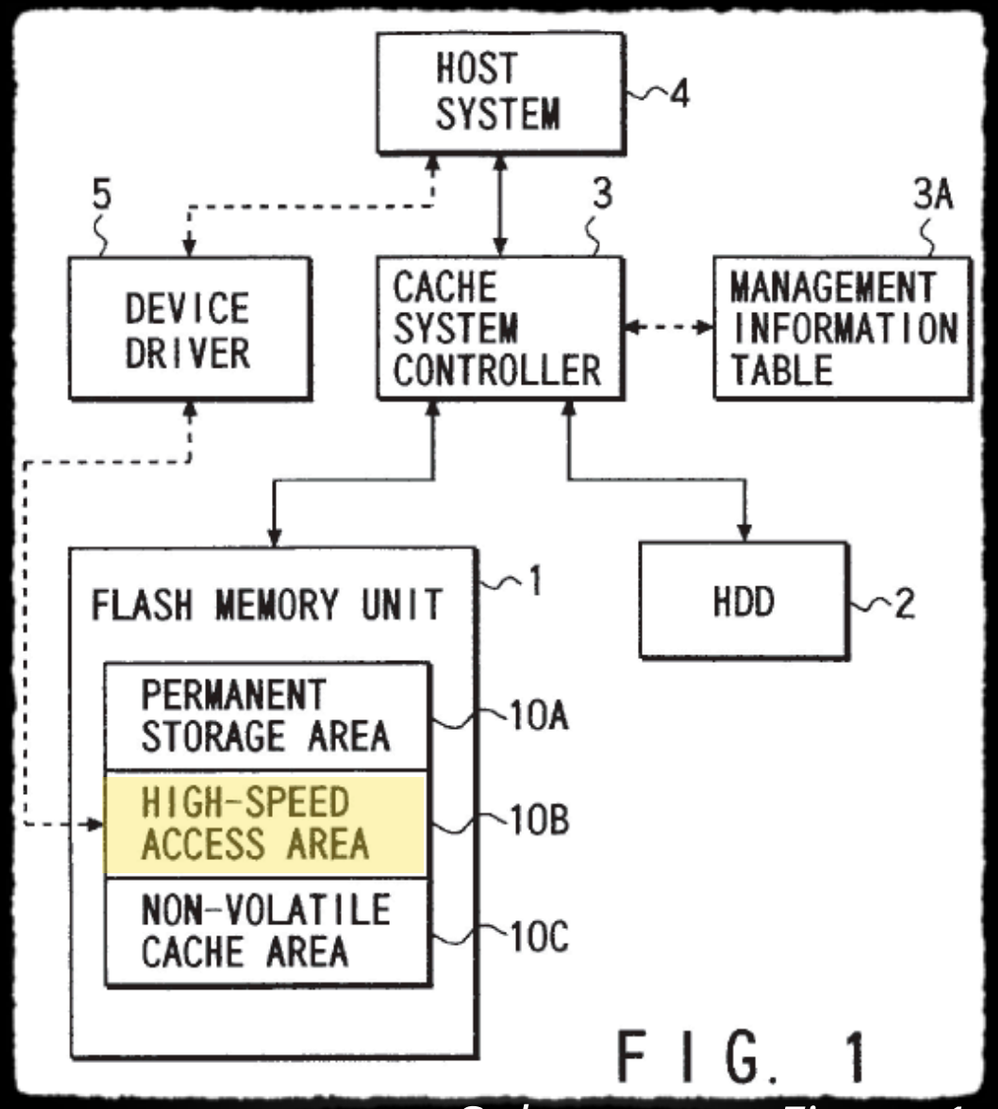
FIG. 1

Sukegawa at Figure 1

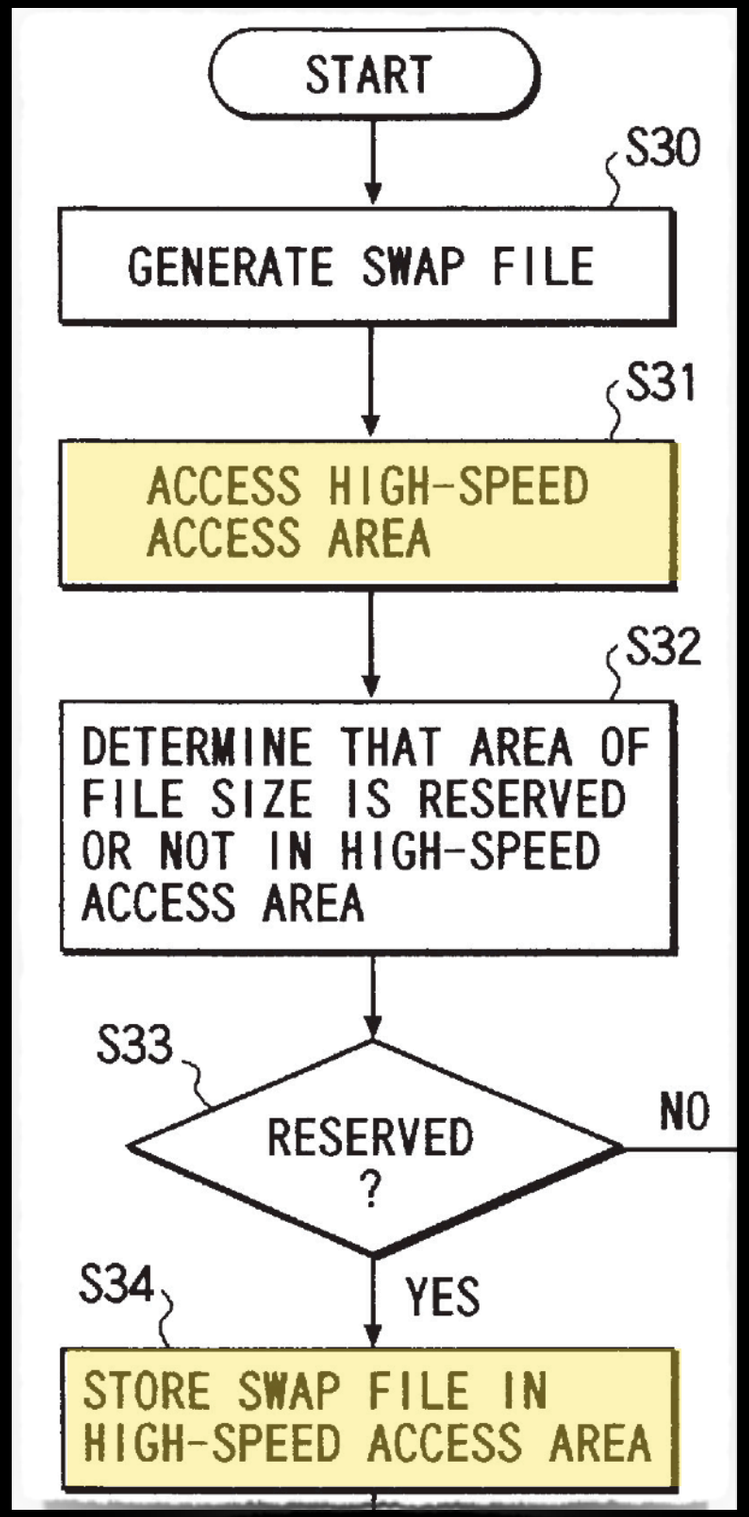
are managed. The controller **3** manages the storage areas **10**A to **10**C of the flash memory unit **1** by using a management information table **3**A. The management information table **3**A is stored, for example, in the non-volatile cache area **10**C of flash memory unit **1**.

Sukegawa at 5:5-9

SUKEGAWA SPECIFIES SWAPPING APPLICATION FILES IN AND OUT OF FLASH AREA 10B—NOT AREAS 10A AND 10C



Sukegawa at Figure 1



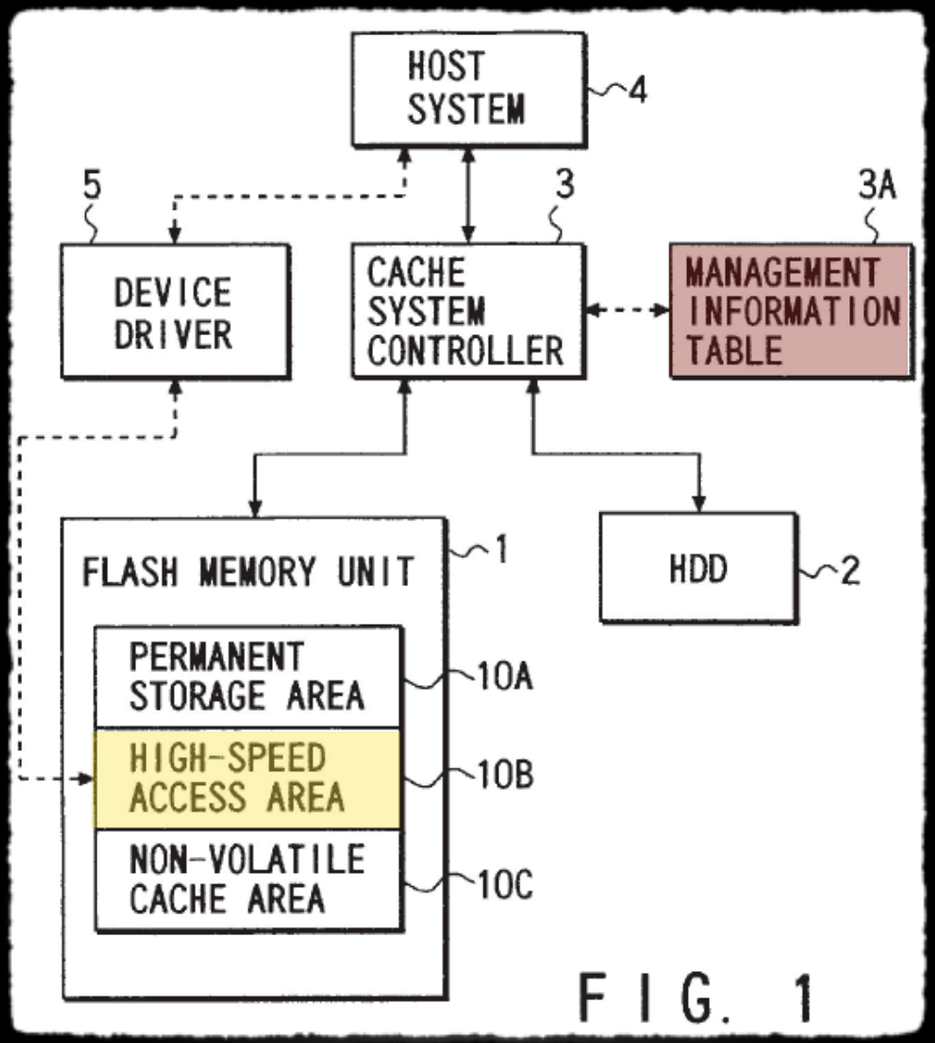
Sukegawa at Figure 6A

NO MOTIVATION EXISTS TO USE ZWIEGINCEW'S SCENARIO FILE WITH SUKEGAWA'S "BOOT DATA LIST"

91. Moreover, Zwiegincew does not provide any teaching or suggestion to add a scenario file to manage Sukegawa's control information files stored in flash memory 1 or to modify the manner in which Sukegawa manages table 3A. As explained above, table 3A is a directory such that no benefit would be gained from teachings of Zwiegincew to manage this directory. Dr. Neuhauser, moreover, does not explain how a POSITA would have incorporated Zwiegincew's teachings toward a scenario file into Sukegawa's table 3A.

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 91

NO MOTIVATION EXISTS TO USE ZWIEGINCEW'S "SCENARIO FILE" WITH SUKEGAWA'S "BOOT DATA LIST"



Sukegawa at Figure 1

92. Sukegawa also discloses a page swapping operation in which application program data is swapped between RAM memory 23 and flash area 10B.[66] However, Sukegawa teaches that table 3A is used for data stored in flash areas 10A and 10C—not flash area 10B.[67] To the extent the teachings of Zwiegincew suggest any modification to Sukegawa's system, a POSITA would have looked to Sukegawa's page swapping operation but not to the management of table 3A. As such, any combination of Sukegawa's and Zwiegincew's teachings does not constitute the claimed "boot data list."

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 92

IPR2016-01737, -01738 ISSUE

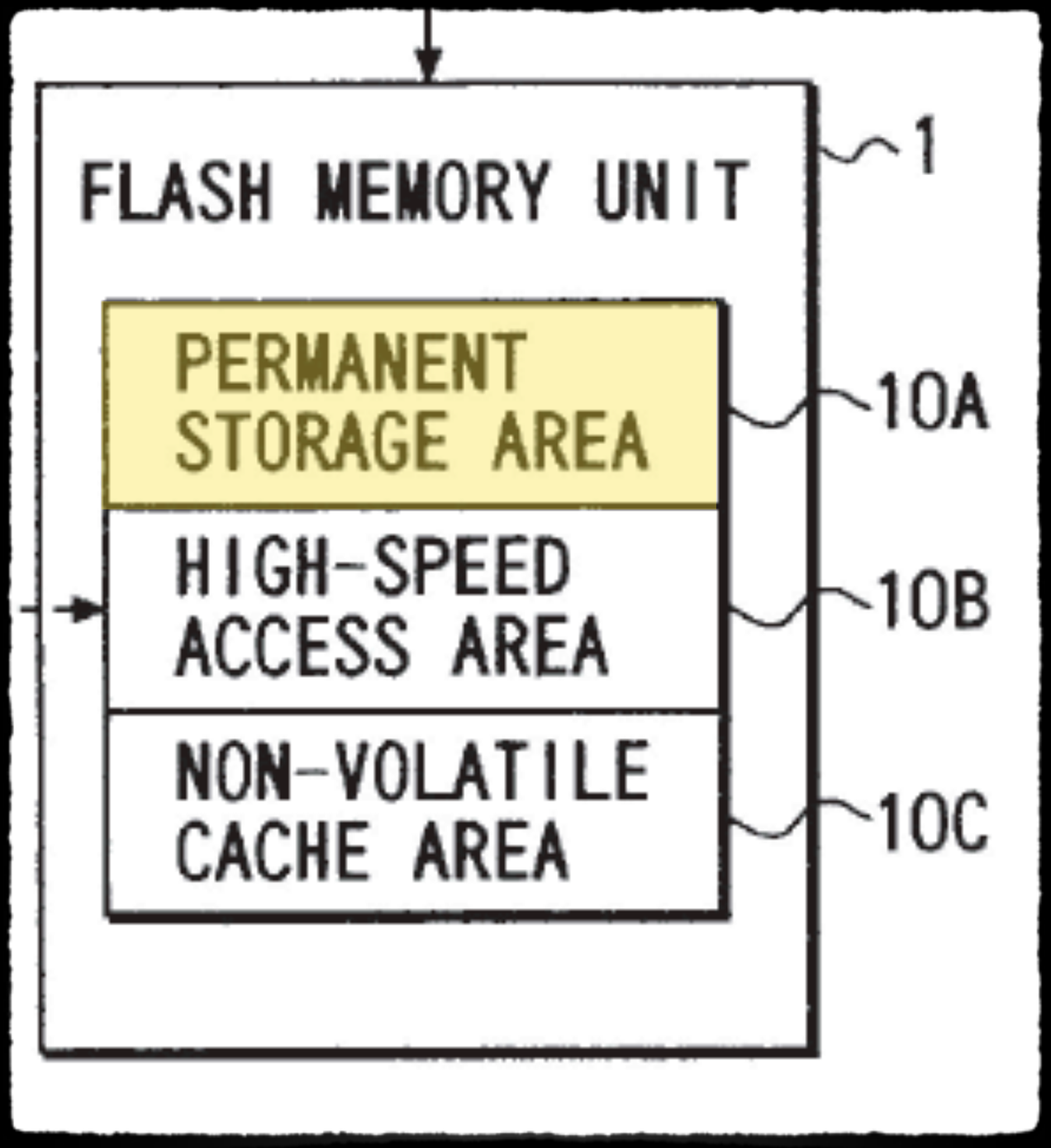
SUKEGAWA DOES NOT DISCLOSE “DISASSOCIATING NON-ACCESSED BOOT DATA FROM THE BOOT DATA LIST”

"NON-ACCESSED BOOT DATA" MEANS "BOOT DATA IDENTIFIED IN THE BOOT DATA LIST THAT WAS NOT REQUESTED DURING SYSTEM BOOT-UP"

66. For purposes of this declaration, it is my opinion that a POSITA would have understood that the term "non-accessed boot data," as used in claims 20, 96, 98, 100, 102, 104, and 106 of the '862 Patent, to mean "boot data identified in the boot data list that was not requested during system boot-up." In my opinion, this is the broadest reasonable interpretation in light of the specification.

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 66

SUKEGAWA DISCLOSES THAT THE OS CONTROL INFORMATION FILE IS SAVED IN FLASH STORAGE AREA 10A



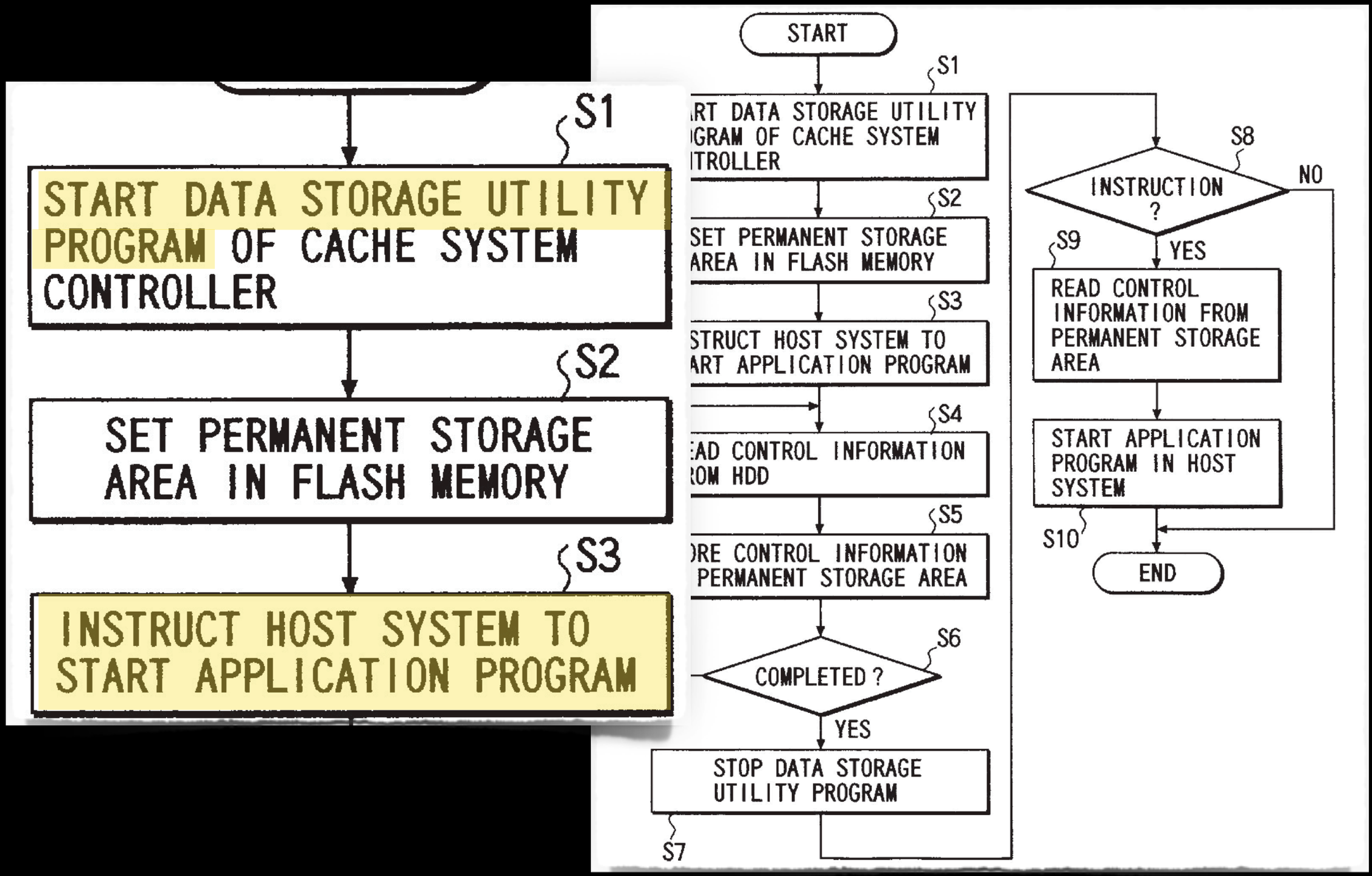
Sukegawa at Figure 1

SUKEGAWA'S MANUAL DELETION OF OS CONTROL INFORMATION FILE IS NOT "DISASSOCIATING NON-ACCESSED BOOT DATA"

98. First, a POSITA would not have understood that a user deleting control information from Sukegawa's flash storage area 10A corresponds to disassociating "non-accessed boot data." Sukegawa discloses that deletion of control information from storage area 10A is "based on the user's judgment."[72] Accordingly, the user can delete control information at any time regardless of whether that information was requested during system boot-up.

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 98

SUKEGAWA USES A USER INTERFACE TO ACCESS APPLICATION PROGRAM CONTROL INFORMATION FILES FROM FLASH



Sukegawa at Fig. 3

SUKEGAWA DOES NOT DISCLOSE "DISASSOCIATING NON-ACCESSED BOOT DATA" ELEMENT

control information files to meet the "disassociating" limitation. Sukegawa discloses that OS control information is only stored in flash storage area 10A,[73] and thus, no OS control information would be stored in flash cache area 10C. As explained above, AP control information is not the claimed "boot data" because Sukegawa discloses AP control information is accessed using a user interface.[74] As such, a POSITA would not have considered the AP control information stored in flash memory 1 as "boot data" or the claimed "non-accessed boot data," in claims 96, 100, and 106.

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 99

MEMORY'S LEAST-RECENTLY-USED ALGORITHM FAILS TO SUGGEST "DISASSOCIATING NON-ACCESSED BOOT DATA" ELEMENT

the "disassociating" limitation. A POSITA would have understood that cache memory using a least-recently-used (LRU) algorithm adds items to cache, and if necessary, discards least-recently-used items. Thus, a POSITA would have understood that an LRU algorithm does not evaluate whether items to be discarded were requested during system boot-up or not. On the contrary, an LRU algorithm could discard items from the cache that were requested during system boot-up if those items happen to be the least-recently-used items when eviction from the cache is taking place. In other words, a cache memory using an LRU algorithm does not "disassociate" items based on whether that item was accessed during boot-up. As such, Sukegawa does not render obvious "disassociating the non-

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 100

IPR2016-01737, -01738 ISSUE

SUKEGAWA DOES NOT DISCLOSE “LOADING [OR ACCESSING] BOOT DATA … THAT IS ASSOCIATED WITH A BOOT DATA LIST”

INDEPENDENT CLAIMS 1, 6, 8, 11, & 13 REQUIRE "LOADING" "BOOT DATA" … THAT IS ASSOCIATED WITH A BOOT DATA LIST"

11. A method for providing accelerated loading of an operating system in a computer system, comprising:

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;

accessing the loaded boot data in compressed form from the memory;

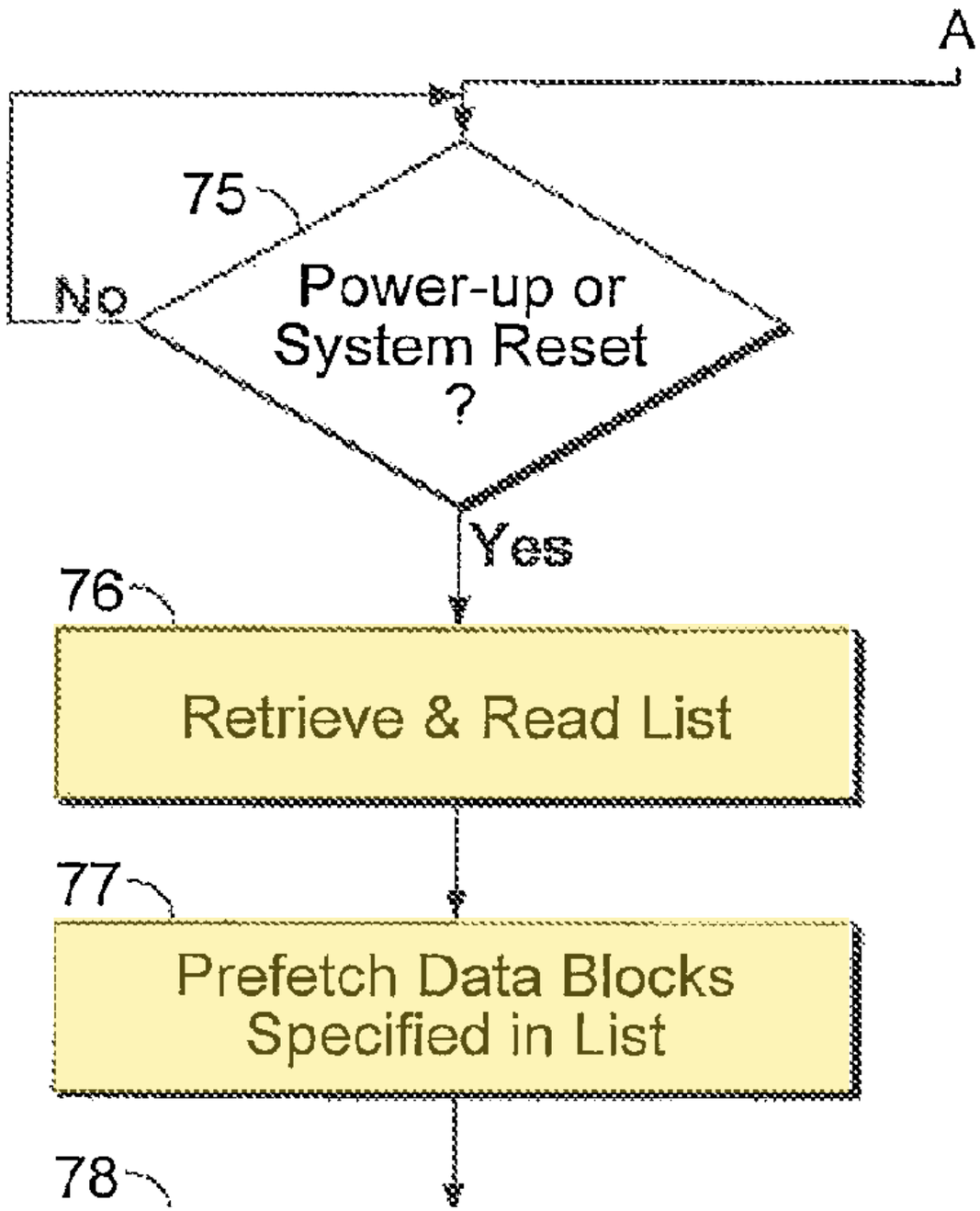
decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;

utilizing the decompressed boot data to load at least a portion of the operating system for the computer system; and

updating the boot data list.

'862 Patent at Claim 11

‘862 SPECIFICATION REQUIRES “BOOT DATA” BE ASSOCIATED WITH “BOOT DATA LIST” PRIOR TO “LOADING” INTO MEMORY



‘862 Patent at Figure 7B

"BOOT DATA" MUST BE ASSOCIATED WITH "BOOT DATA LIST" PRIOR TO "LOADING" INTO MEMORY

102. A POSITA would have understood that claims 1, 6, and 13 require that the recited "boot data" be "associated with a boot data list" *prior to* the boot data being loaded into memory. This requirement is consistent with the specification of the '862 Patent. The specification consistently describes that "boot data" be associated with the "boot data list" at the time the boot data is loaded into cache memory.[75] In fact, the system's storage controller uses the boot data list to identify which data is the boot data to be loaded into memory.[76] For example, the

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 102

"BOOT DATA" MUST BE ASSOCIATED WITH "BOOT DATA LIST" PRIOR TO "LOADING" INTO MEMORY

103. This requirement is further supported by the distinction the '862 claims make between boot data "associated" and "not associated" with the boot data list. For example, claim 1 specifies boot data "that is associated with a portion of a boot data list," and claim 6 and 13 include similar limitations. In contrast, claims 99 and 105 specify accessing boot data "that is not associated with the boot data list." In view of this distinction, a POSITA would have understood that the boot data in claims 1, 6, and 13 must already be associated with the boot data list before being loaded into memory.

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 103

DR. NEUHAUSER ADMITS CLAIMS REQUIRE "BOOT DATA" BE ASSOCIATED WITH "BOOT DATA LIST" PRIOR TO "LOADING"

Q The loading step of Claim 1 specifies that the boot data being loaded is associated with a portion of a boot data list, correct?

A I believe that's correct.

So the boot data being loaded from a boot device is both in a compressed form and associated with a boot data list, correct?

A Yes. And I think it's both of those things. That's correct.

Dr. Neuhauser Transcript (Ex. 2011) at 25:21-26:16, 38:9-18

DR. NEUHAUSER ADMITS CLAIMS REQUIRE "BOOT DATA" BE ASSOCIATED WITH "BOOT DATA LIST" PRIOR TO "LOADING"

Q So Claim 6, from the perspective of a person of ordinary skill in the art, understands that the processor is configured to load a portion of boot data -- of the boot data, correct?

A That's correct.

Q And a person of -- a POSITA understands that the portion of boot data that's loaded is in the compressed form and is associated with a boot data list, correct?

MR. BITTNER: Objection; form.

A Yes.

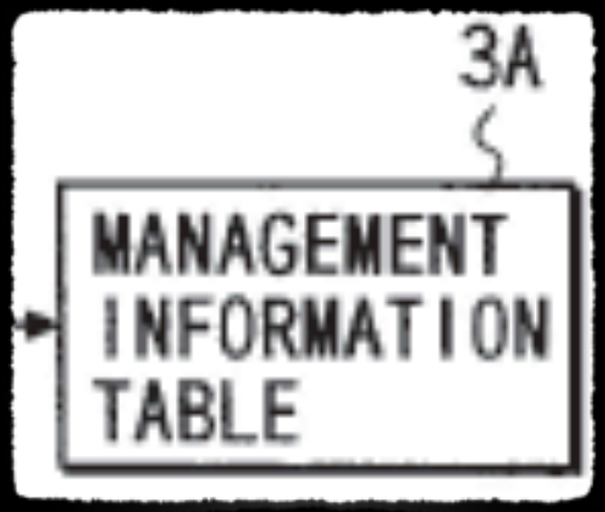
THE WITNESS: Read the question again.

(Requested portion read.)

A Yes, I believe that's correct.

Dr. Neuhauser Transcript (Ex. 2011) at 41:4-16

APPLE ASSERTS SUKEGAWA'S MANAGEMENT INFORMATION TABLE 3A IS THE CLAIMED "BOOT DATA LIST"



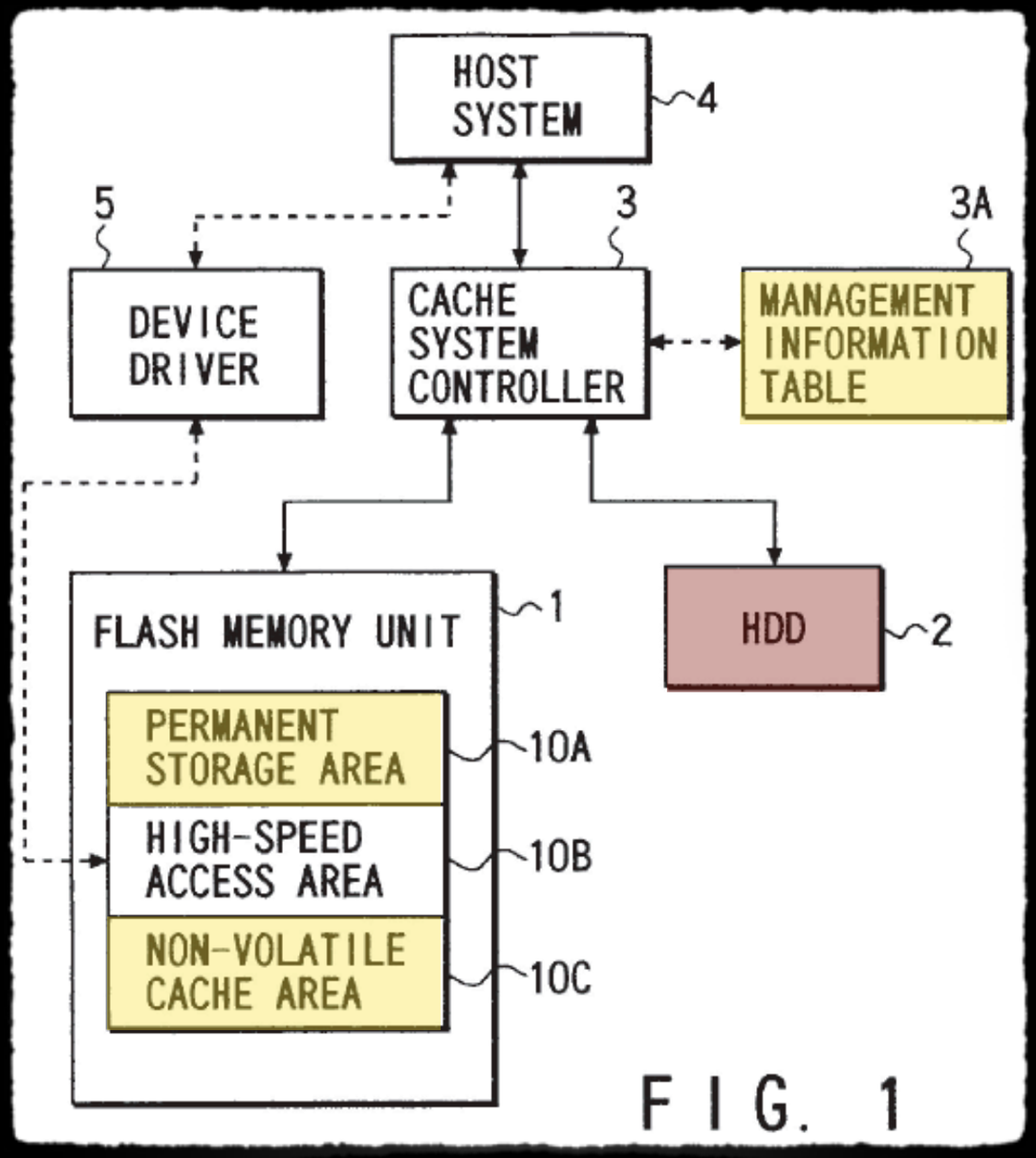
Sukegawa at Figure 1

memory unit 1." Id., 5:1-61, 6:59-7:2. Thus, Sukegawa's table 3A stores information descriptive of the data loaded into the storage areas 10A/10C of the flash memory 1 (e.g., files of application/OS control information). Because a table includes a list of data (e.g., rows of data stored in the table), because Sukegawa's storage areas 10A/10C are loaded with boot data, and because Sukegawa's table 3A identifies the boot data stored in storage areas 10A/10C, Sukegawa's table 3A includes a boot data list (e.g., a list of files of application/OS data). Dec., ¶¶125-126.

'1737 Petition at 12

SUKEGAWA'S TABLE 3A NEVER LOADS "BOOT DATA … THAT IS ASSOCIATED WITH A BOOT DATA LIST"

SUKEGAWA'S TABLE 3A ONLY MANAGES FLASH MEMORY UNITS 10A AND 10C—NOT HARD DRIVE 2:



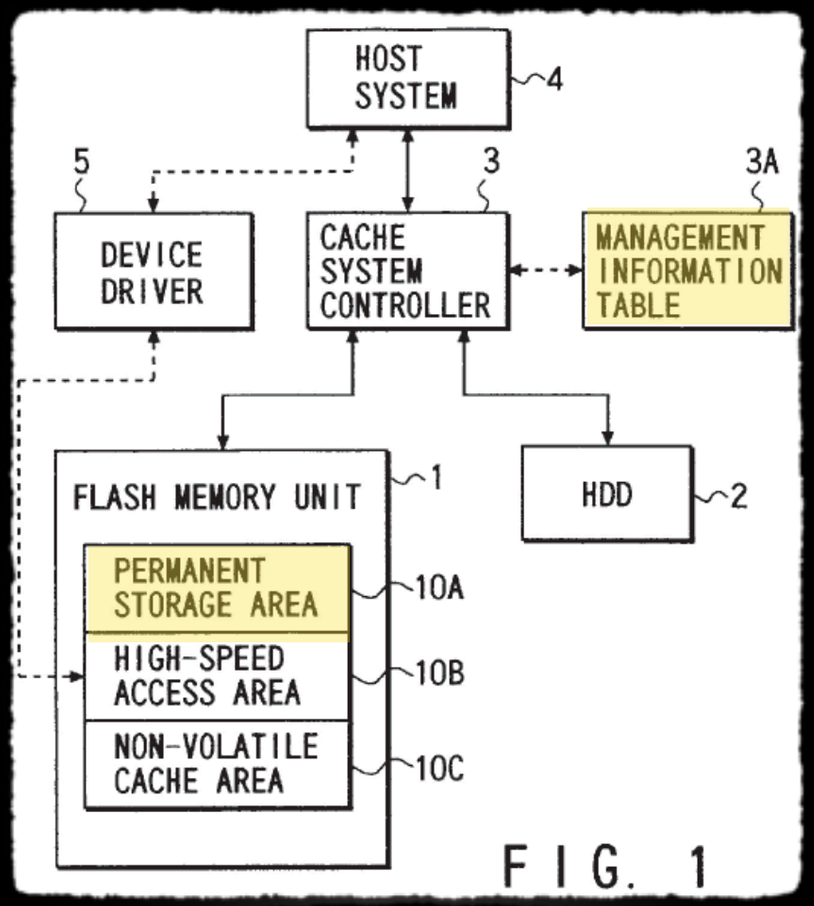
Sukegawa at Figure 1

are managed. The controller **3** manages the storage areas **10**A to **10**C of the flash memory unit **1** by using a management information table **3**A. The management information table **3**A is stored, for example, in the non-volatile cache area **10**C of flash memory unit **1**.

Sukegawa at 5:5-9

SUKEGAWA'S TABLE 3A NEVER LOADS "BOOT DATA … THAT IS ASSOCIATED WITH A BOOT DATA LIST"

SUKEGAWA'S ALLEGED "BOOT DATA" (CONTROL INFORMATION FILES) IS ASSOCIATED WITH TABLE 3A ONLY AFTER THE DATA HAS BEEN LOADED INTO FLASH FROM THE BOOT DEVICE:



Sukegawa at Figure 1

In this case, the control information is stored in the permanent storage area **10**A in flash memory unit **1** under the file name designated by the user. Information for correlating the file name and the AP and information of other comment is recorded on the management information table **3**A by the data storage utility program. The user inputs the file name to the controller **3** via the user interface, thereby referring to the file (the control information of the AP in this case) stored in the permanent storage area **10**A. The user can delete the file,

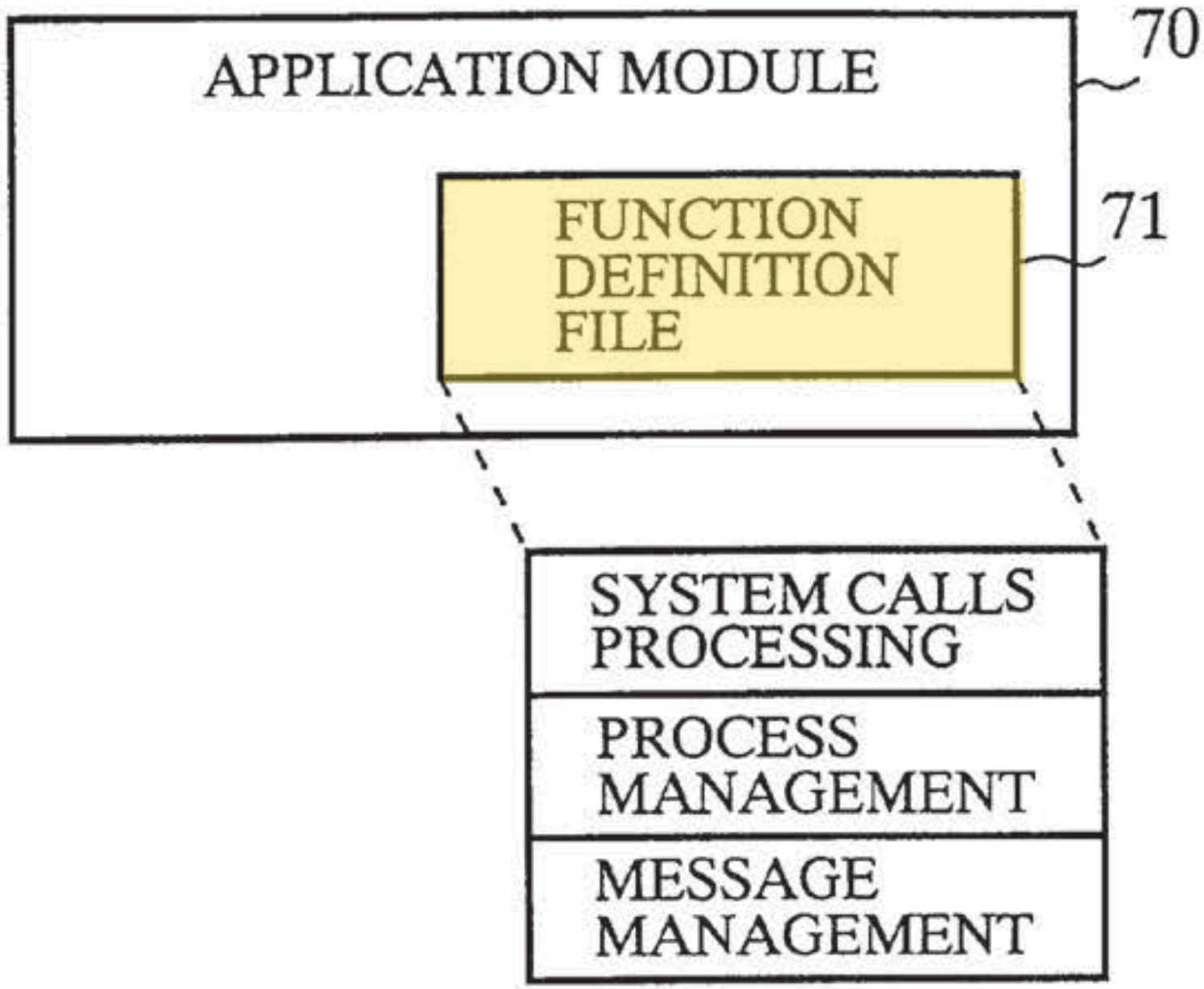
Sukegawa at 5:41-53

SUKEGAWA'S TABLE 3A NEVER LOADS "BOOT DATA … THAT IS ASSOCIATED WITH A BOOT DATA LIST"

information is recorded in table 3A.[80] Assuming for the sake of argument that Sukegawa's table 3A were "a boot data list," then the boot data Sukegawa loads into its cache does not become associated with that list until after it has been loaded into the cache, not before. This is so because Sukegawa's system updates its management information table only when loading data into the cache. Thus, Sukegawa fails to disclose that its boot data "is associated with a boot data list" (table 3A) prior to being loaded into flash memory 1, as required by claims 1, 6, and 13.

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 107

SETTSU'S FUNCTION DEFINITION FILE IS NOT RELATED TO TABLES LIKE SUKEGAWA'S TABLE 3A



Settsu at Figure 18

SETTSU DOES NOT RENDER OBVIOUS "LOADING BOOT DATA … THAT IS ASSOCIATED WITH A BOOT DATA LIST"

directorial operation of Sukegawa's' table 3A. Specifically, Settsu's function definition file 71, which is part of application module 70, lists between one and seven functional modules drawn from the modules contained in OS main body 5 for loading into memory.[83] A POSITA would not have considered Settsu's teachings regarding function definition file 71 pertinent to the operation of Sukegawa's table 3A or to storage of OS and AP control information in Sukegawa's permanent storage area 10A.

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 109

ZWIEGINCEW'S SCENARIO FILE IS NOT RELATED TABLES LIKE SUKEGAWA'S TABLE 3A

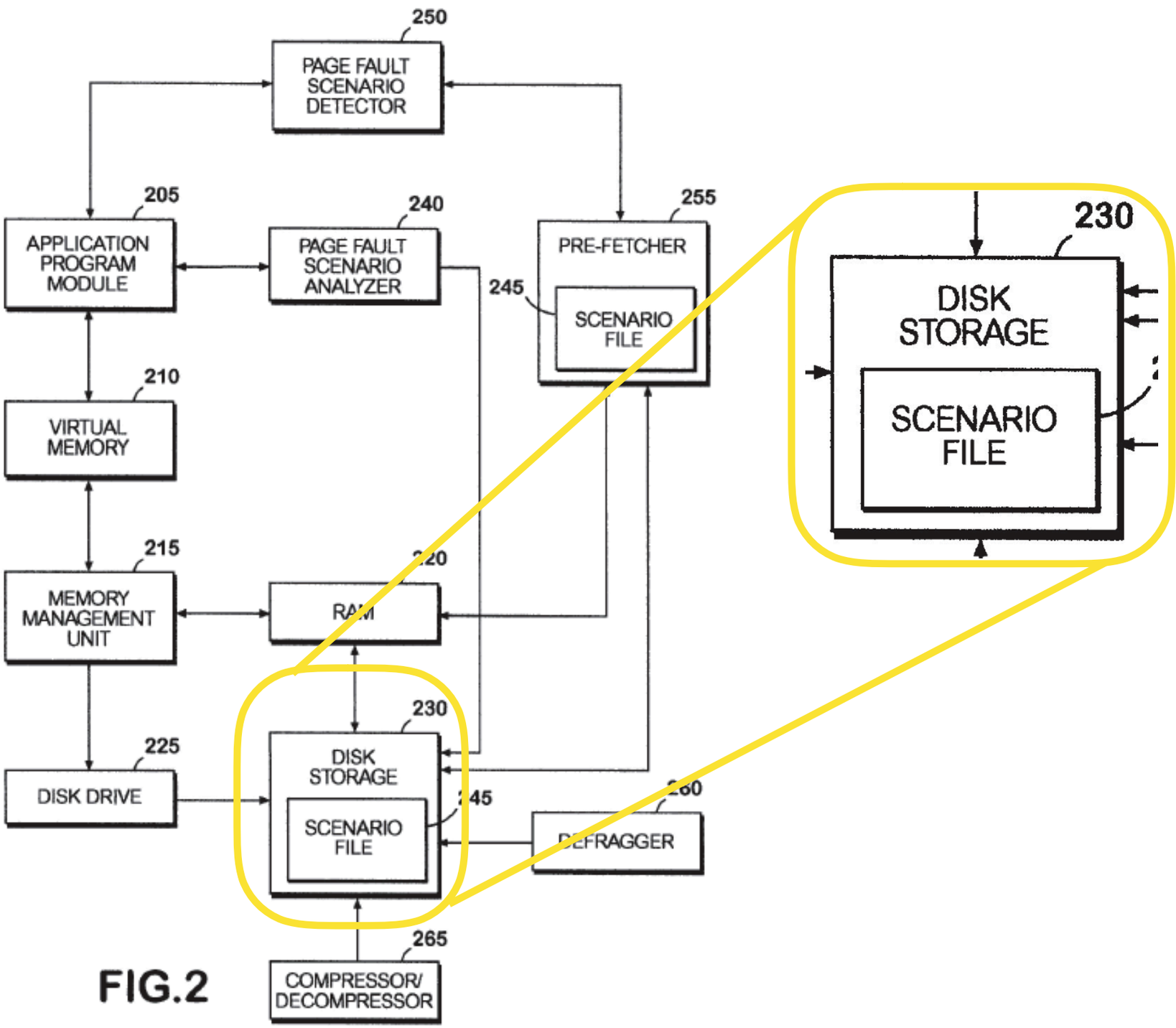


FIG.2

Zwiegincew at Figure 2

ZWIEGINCEW DOES NOT RENDER OBVIOUS "LOADING BOOT DATA … THAT IS ASSOCIATED WITH A BOOT DATA LIST"

111. As explained above in Section VII.B., Dr. Neuhauser mischaracterizes Zwiegincew's teachings. A POSITA would not have considered Zwiegincew's scenario file to be a "form of boot data list" or to use the scenario during the boot process.[85] As such, a POSITA would not have considered Zwiegincew's teachings regarding the scenario file pertinent to the operation of Sukegawa's table 3A or to the storage of OS and AP control information into permanent storage area 10A.

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 111

CLAIM 14 REQUIRES "ACCESSING BOOT DATA … ASSOCIATED WITH A BOOT DATA LIST"

14. A method for providing accelerated loading of an operating system in a computer system, comprising:

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;

loading the boot data into a memory; and

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; and

updating the boot data list.

'862 Patent at Claim 14

LIKE "LOADING" ELEMENT, SUKEGAWA DOES NOT DISCLOSE CLAIM 14'S "ACCESSING BOOT DATA … THAT IS ASSOCIATED WITH A BOOT DATA LIST"

APPLE ARGUES THAT SUKEGAWA ALONE DISCLOSES CLAIM 14'S "ACCESSING" STEP

107. Sukegawa does not disclose the alleged "boot data" is associated with the alleged "boot data list" before "accessing" and "loading" the boot data into flash memory 1. As explained above in Section VII.A., Apple and Dr. Neuhauser assert that Sukegawa discloses a "boot data list" under two theories. However, Sukegawa's AP and OS control information (i.e., the alleged "boot data") is not "associated with a boot data list" prior to accessing the data in HDD 2 nor loading the data from HDD 2 to flash memory 1.

Dr. Back '1738 Declaration (Ex. 2008) at ¶ 107

IPR2016-01737, -01738 ISSUE

SUKEGAWA DOES NOT DISCLOSE CLAIM 14’S “ACCESSING BOOT DATA” PRIOR TO “LOADING”

CLAIM 14 DIFFERENTIATES BETWEEN ACCESSING UNLOADED BOOT DATA AND ACCESSING LOADED BOOT DATA

14. A method for providing accelerated loading of an operating system in a computer system, comprising:

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;

loading the boot data into a memory; and

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; and

updating the boot data list.

'862 Patent at Claim 14

CLAIM 14 REQUIRES "ACCESSING BOOT DATA" PRIOR TO "LOADING THE BOOT DATA"

100. Claim 14 recites three steps of "accessing boot data," "loading the boot data," and "servicing a request for the boot data." A POSITA would have understood that claim 14 requires initially "accessing boot data," then "loading the boot data" that was accessed, and subsequently "servicing a request for the boot data…to access the loaded…boot data." As such, a POSITA would have understood that the "accessing" step is the accessing of ***unloaded*** boot data, whereas the "servicing" step is to access and decompress ***loaded*** boot data. In other words, claim 14 requires the "accessing" and "servicing" steps to each interact with boot data in different forms, and at different stages of the boot process. Consequently, the "accessing" and "servicing" steps cannot comprise the same step of the boot process.

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 100

APPLE ASSERTS SUKEGAWA "ACCESSING BOOT DATA" ONLY AFTER IT HAS BEEN LOADED INTO FLASH MEMORY 1

14.1: 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;

As explained at 8.0-8.3 and 11.0-11.2, Sukegawa and Dye render obvious loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory, and accessing the loaded boot data in compressed form from the memory. Dec., ¶¶428-430.

Similarly, and as Dr. Neuhauser explains, Sukegawa and Dye render obvious 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. Dec., ¶¶431-432.

'1738 Petition at 31-32

SUKEGAWA'S HOST SYSTEM ACCESSES ALLEGED "BOOT DATA" AFTER THAT DATA HAS BEEN LOADED INTO FLASH MEMORY 1

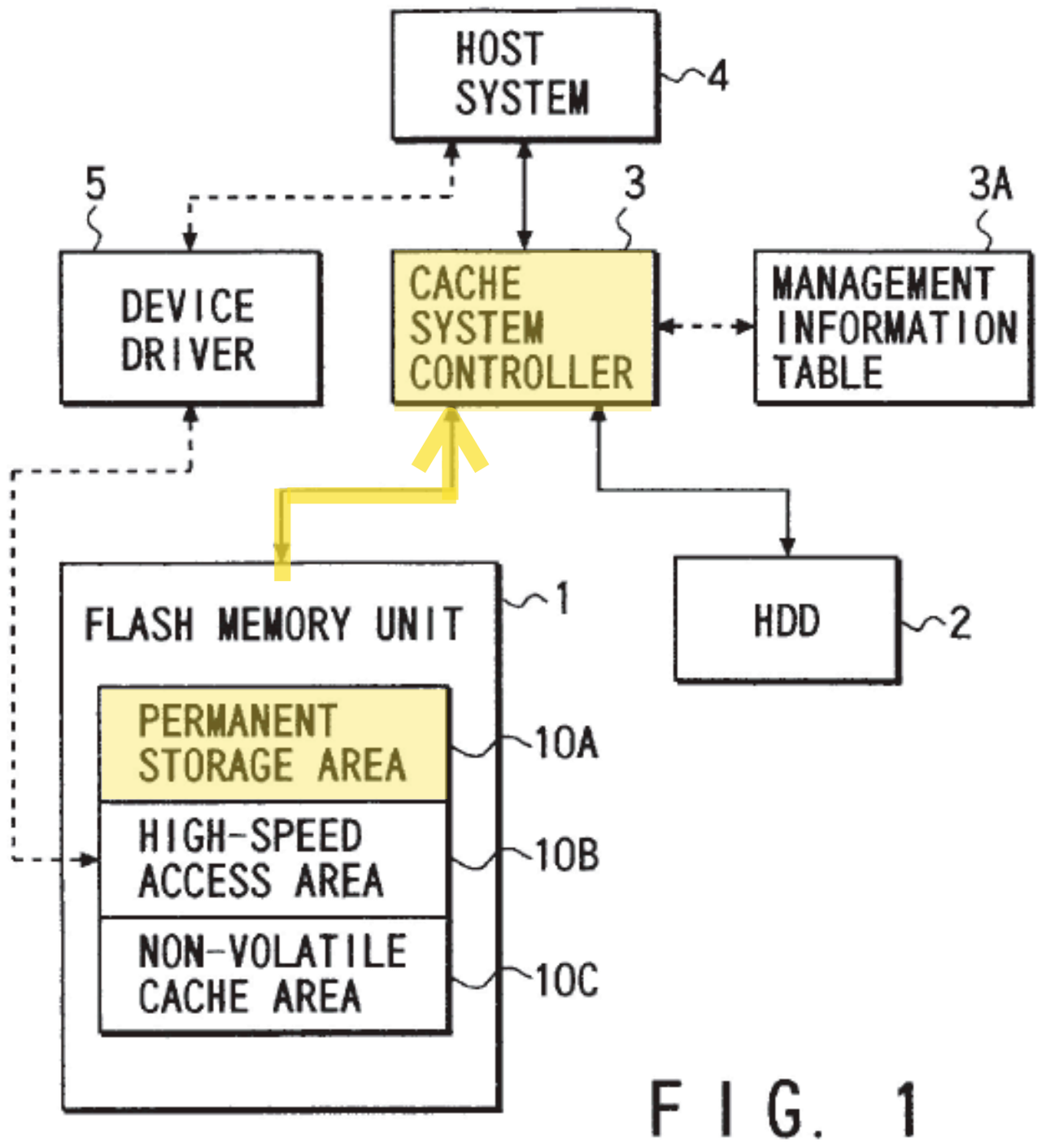


FIG. 1

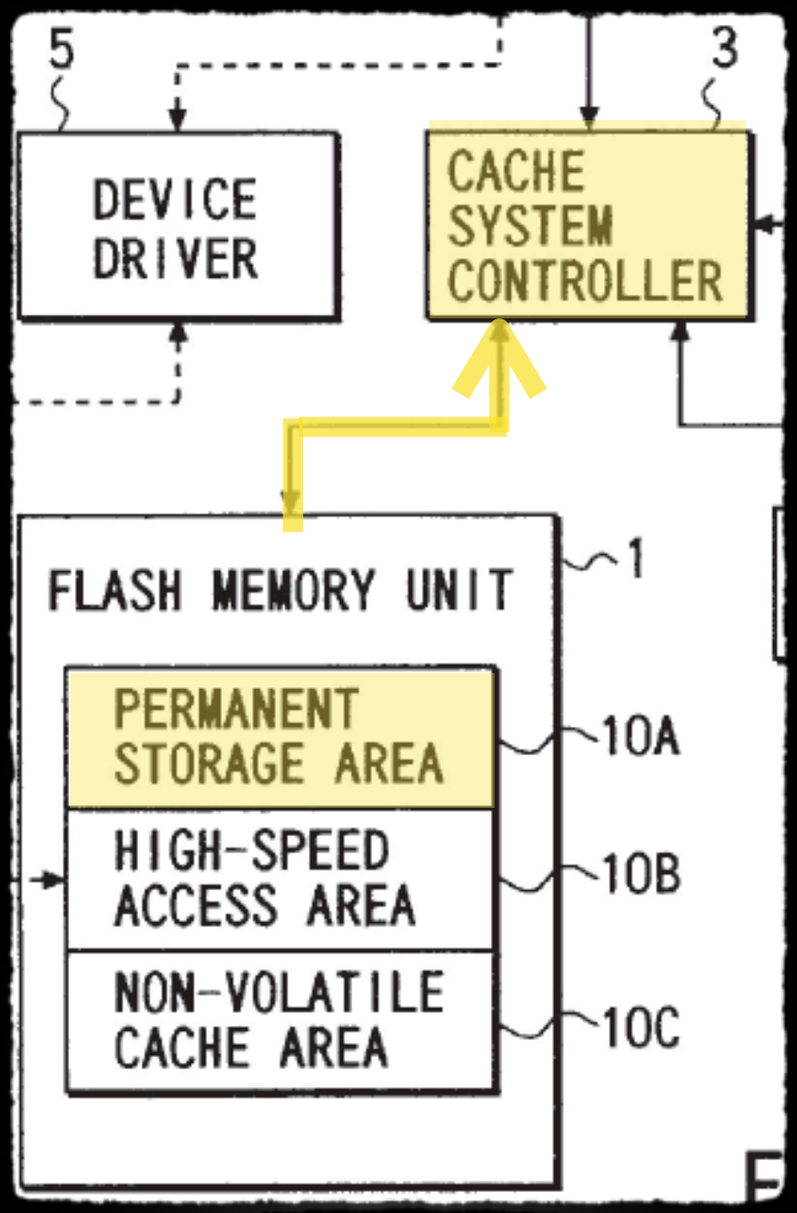
Sukegawa at Figure 1

DR. NEUHAUSER CONCEDES SUKEGAWA ONLY ACCESSES BOOT DATA AFTER IT HAS BEEN LOADED INTO FLASH MEMORY 1

compression/decompression engine. With this modification, Sukegawa's controller 3 loads control information into the flash memory unit 1 in compressed form, and uses the compressed control information to service requests from Sukegawa's host system. In doing so, Sukegawa's controller 3 accesses, from flash memory unit 1, the compressed control information and uses Dye's compression/decompression engine to decompress the compressed data accessed from flash memory unit 1 at a rate that increases flash memory unit 1's effective access rate [Dye '284, 4:16-20, 11:32-35, 11:56-12:7, 12:61-13:7, 13:52-53; Dye,

Dr. Neuhauser '1738 Declaration (Ex. 1003) at ¶ 429

DR. NEUHAUSER CONCEDES SUKEGAWA ONLY ACCESSES BOOT DATA AFTER IT HAS BEEN LOADED INTO FLASH MEMORY 1



Sukegawa at Figure 1

Thus, one having ordinary skill would have found it obvious to modify Sukegawa's controller 3 to use an embedded compression/decompression engine, and to service requests from Sukegawa's host system by accessing compressed data from flash memory unit 1 and decompressing the compressed data.

Dr. Neuhauser '1738 Declaration (Ex. 1003) at ¶ 430

SUKEGAWA FAILS TO DISCLOSE CLAIM 14'S "ACCESSING BOOT DATA" PRIOR TO "LOADING" AND "SERVICING" STEPS

102. Accordingly, Sukegawa fails to disclose the separate "accessing" and "servicing" steps because Sukegawa only accesses the control information (the alleged "boot data") after that data has been loaded into memory. Sukegawa, therefore, does not disclose both the "accessing" and "servicing" steps of claim 14.

Dr. Back '1738 Declaration (Ex. 2008) at ¶ 102

IPR2016-01737, -01738 ISSUE

SUKEGAWA DOES NOT DISCLOSE CLAIM 19’S “UTILIZING THE STORED ADDITIONAL PORTION OF [OS]”

‘862 CLAIM 9 REQUIRES UTILIZING STORED ADDITIONAL OS DATA TO FURTHER PARTIALLY BOOT THE SYSTEM (‘1738 IPR)

9. The method of claim **8**, further comprising:

compressing an additional portion of the operating system that is not associated with the boot data list; and

storing the additional portion of the operating system in the first memory, and

wherein the utilizing comprises:

utilizing the stored additional portion of the operating system to at least further partially boot the computer system.

‘862 Patent at Claim 9

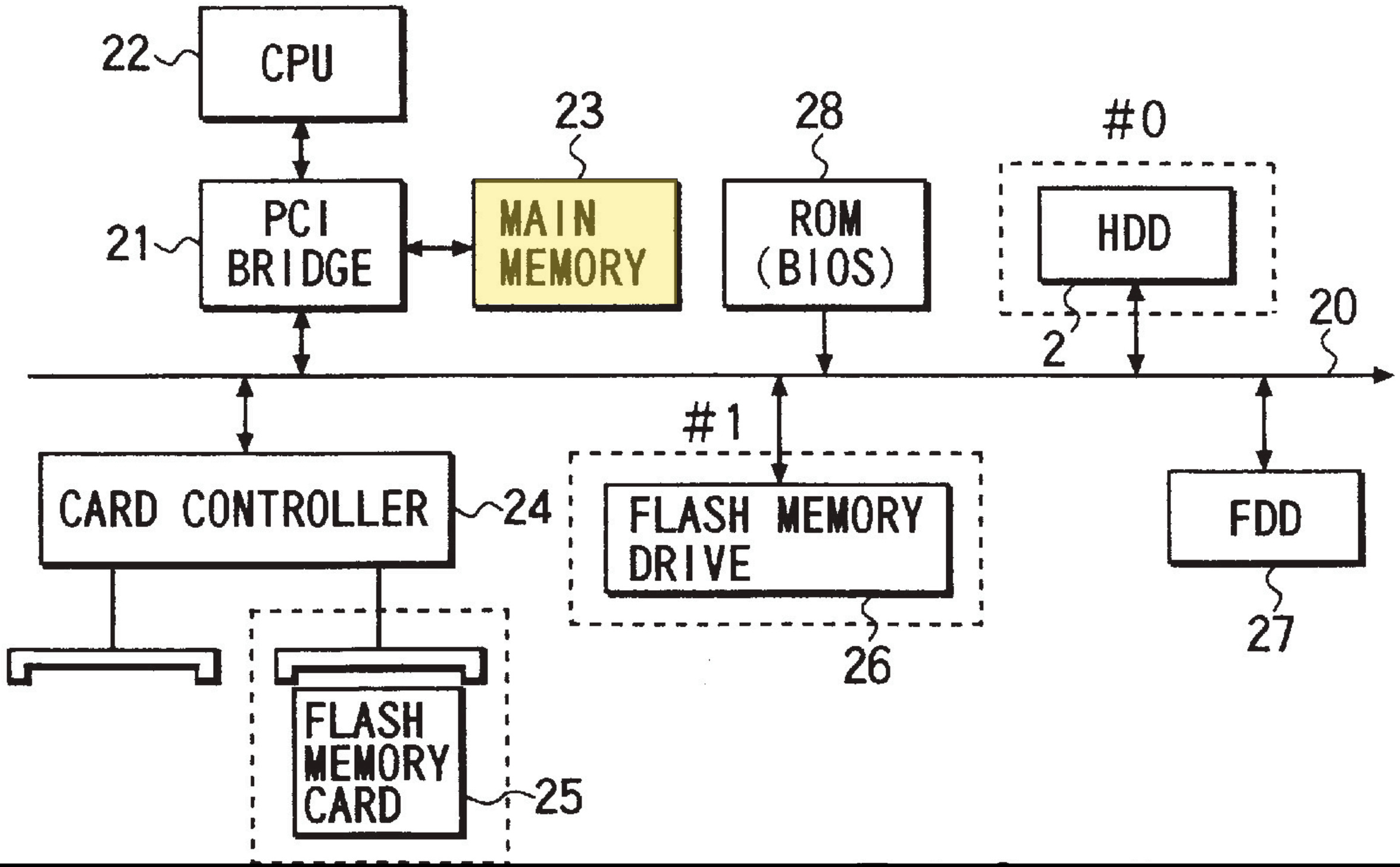
APPLE ASSERTS SUKEGAWA'S FILE SWAPPING MEETS CLAIM 9'S "UTILIZING" STEP

9.2: … utilizing the stored additional portion of the operating system to at least further partially boot the computer system.

As explained at 8.1-8.7 and 9.0-9.1, Sukegawa discloses storing a swap file that includes an additional portion of the OS that is necessary for starting the OS on HDD 2. Sukegawa, 7:40-55, 7:66-8:6, 8:17-36; Dec., ¶373. Sukegawa further discloses that the swap file can be read out via controller 3 on an as-needed basis, and a POSITA would have understood that the stored additional portion of the OS in the swap file could be used to further partially boot Sukegawa's host computer system. Sukegawa, 4:26-30, 5:10-40, 6:19-58, 7:28-55, 8:17-18, 8:21-36.

'1738 Petition at 24

SUKEGAWA SWAPS APPLICATION PROGRAM DATA IN AND OUT OF MAIN MEMORY RAM 23 AFTER SYSTEM BOOT-UP



Sukegawa at Figure 2

SUKEGAWA SWAPS APPLICATION PROGRAM DATA IN AND OUT OF MAIN MEMORY RAM 23 AFTER SYSTEM BOOT-UP

of the HDD. In a specific system, a part of the storage area of the main memory (volatile IC memory) comprising a DRAM is used as a cache area of the HDD (this system being called "smartdrive"). In this system, however, the

Sukegawa at 1:20-24

When the host system **4** executes a program other than the programs such as the AP stored in the main memory **23**, the host system **4** performs a swapping operation. In the swapping operation, the program (including data) stored in the main memory **23** is shifted from the main memory **23** as a swap file, thereby to load the program to be executed in the main memory **23**. Normally, the swap file is shifted to the HDD **2**.

Sukegawa at 7:66-8:6

SUKEGAWA'S FILE SWAPPING DOES NOT RELATE TO CLAIM 9'S "UTILIZING" STEP

134. First, Sukegawa discloses that any OS control information stored in flash memory 1 is stored as a single file in the permanent storage area 10A.[109] Sukegawa also discloses that its swapping operation swaps application program data between RAM memory 23 and flash area 10B—not flash storage areas 10A and 10C.[110] Accordingly, OS control information stored in flash memory 1 is not ever subject to a swapping operation. A POSITA, thus, would not have understood that any of Sukegawa's swap files contain a "portion of the operating system to at least further partially boot the computer system," as recited in claim 9.

Dr. Back '1738 Declaration (Ex. 2008) at ¶ 134

SUKEGAWA'S FILE SWAPPING DOES NOT RELATE TO CLAIM 9'S "UTILIZING" STEP

135. Moreover, as explained above in Section VII.A., a POSITA would not have considered Sukegawa's AP control information stored in flash memory 1 to be a "portion of the operating system to at least further partially boot the computer system," as recited in claim 9. Sukegawa discloses that a user instruction, issued via a user interface, causes the reading out of AP control information from flash memory 1.[111] To the extent any AP data is subject to Sukegawa's swapping operation, that data is not a "portion of the operating system to at least further partially boot the computer system."

Dr. Back '1738 Declaration (Ex. 2008) at ¶ 135

IPR2016-01737, -01738 ISSUE

SUKEGAWA DOES NOT DISCLOSE “BOOT DATA” WITH “PROGRAM CODE ASSOCIATED WITH … AN APPLICATION PROGRAM”

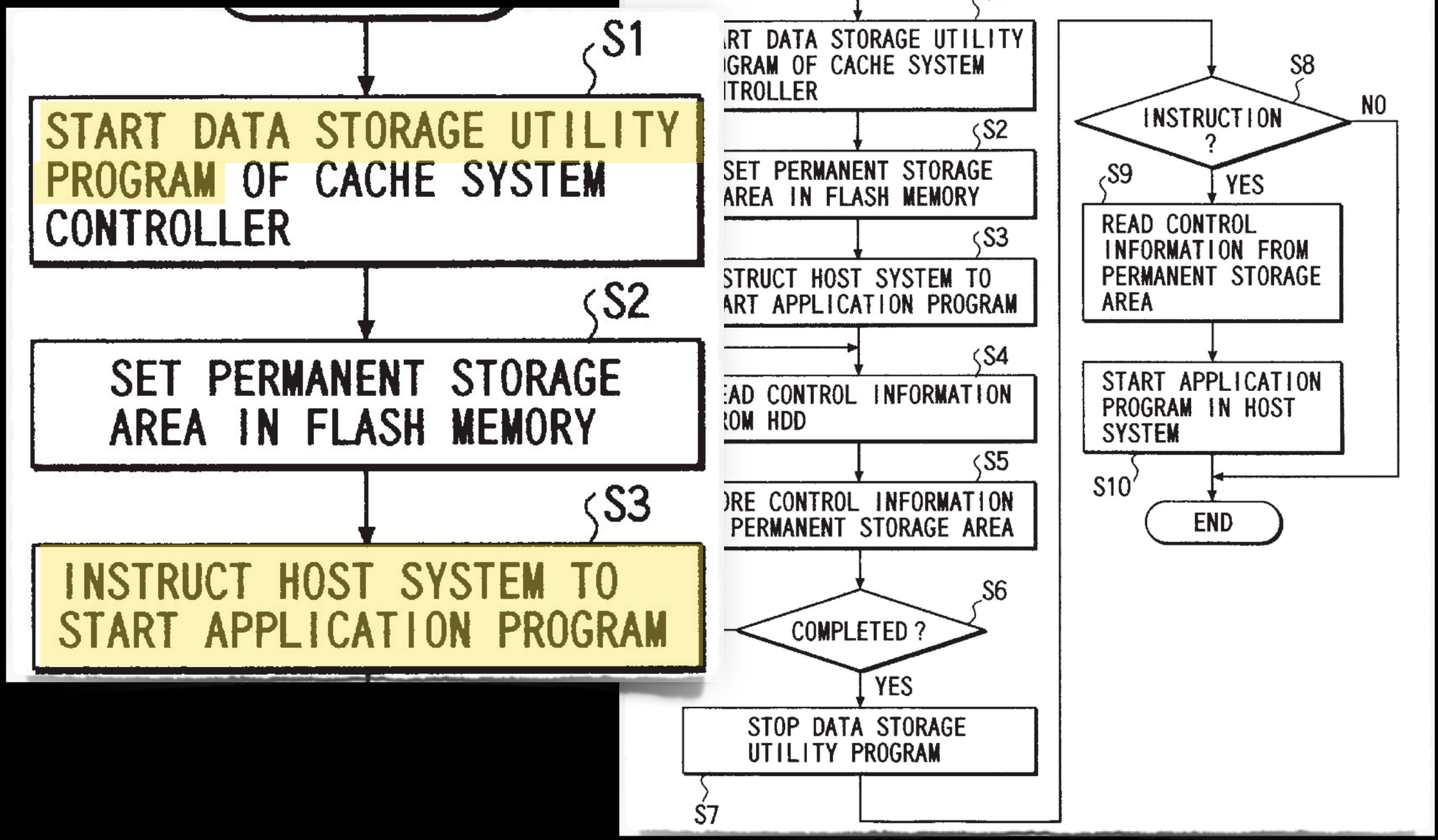
‘862 CLAIMS 17, 29, 53, 65, 77, AND 89 REQUIRE THE “BOOT DATA” INCLUDE PROGRAM CODE ASSOCIATED WITH AN APPLICATION PROGRAM (‘1737, ‘1738 IPRS)

17. The method of claim **14**, wherein the boot data comprises:

a program code associated with the operating system and an application program.

‘862 Patent at Claim 17

SUKEGAWA USES A USER INTERFACE TO ACCESS APPLICATION PROGRAM CONTROL INFORMATION FILES FROM FLASH



Sukegawa at Fig. 3

SUKEGAWA'S APPLICATION PROGRAM CONTROL INFORMATION IS NOT "BOOT DATA"

table 3A references information regarding AP control information files. Because Sukegawa discloses that "the user instructs the start of the same AP [stored in flash] via the user interface,"[57] a POSITA would have understood that Sukegawa's user interface is available to the user only after the system has booted-up. Accordingly, a POSITA would not have considered AP control information files stored in flash memory 1 to be "boot data." Thus, a POSITA would not have understood that table 3's reference to information regarding AP control information in Sukegawa meets the claimed "boot data list."

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 82

SUKEGAWA'S "BOOT DATA" DOES NOT INCLUDE CODE ASSOCIATED WITH AN APPLICATION PROGRAM

limitation those files include program code.[102] But as explained in Section VII.A., a POSITA would not have understood that AP control information stored on Sukegawa's flash memory 1 is "boot data" because that control information is accessed by the user only after completion of the system's boot-up process.[103] As such, a POSITA would not have understood that Sukegawa discloses "boot data" stored in flash memory 1 that includes program code associated with an application program.

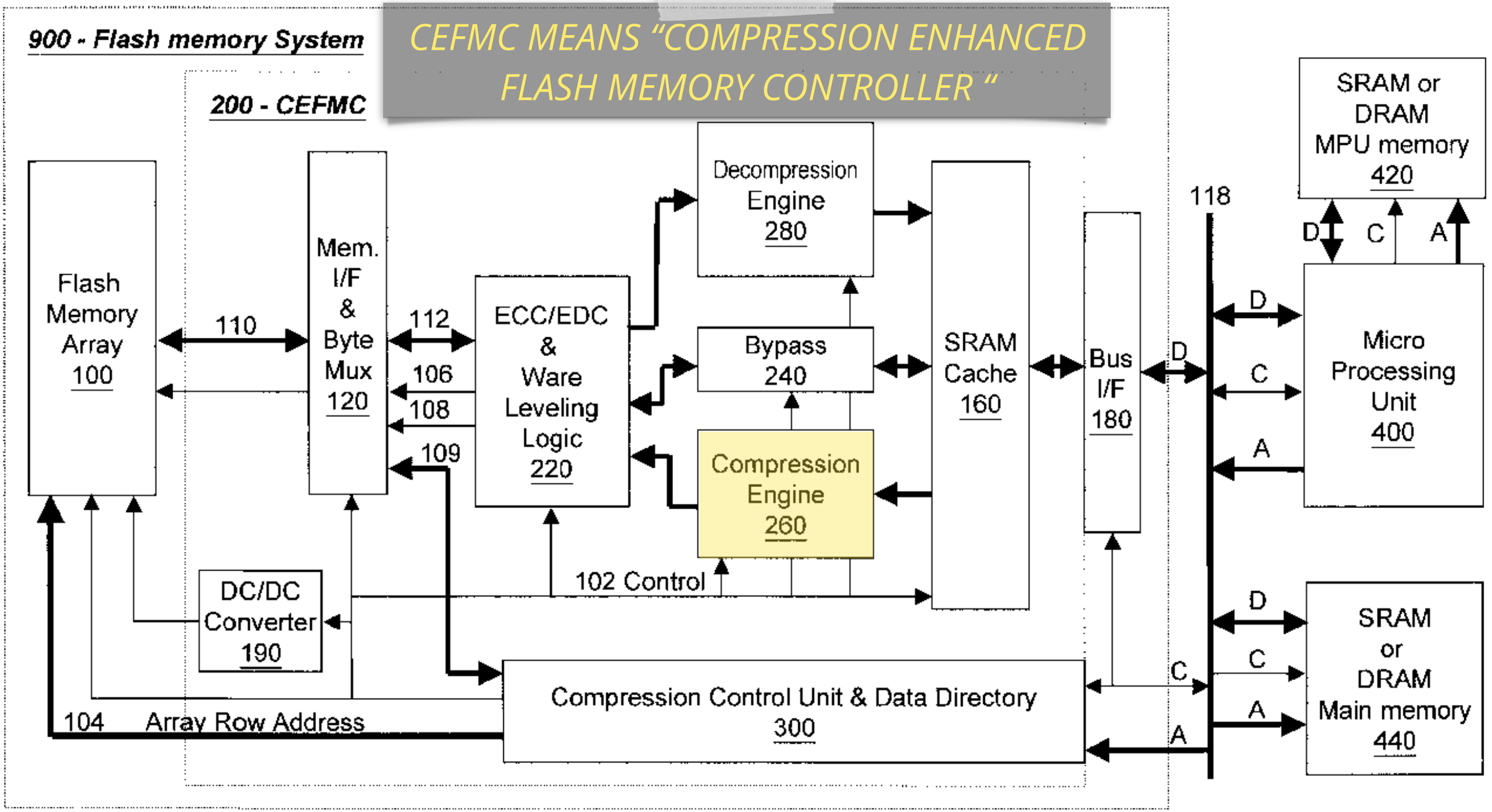
Dr. Back '1737 Declaration (Ex. 2008) at ¶ 124

IPR2016-01737, -01738 ISSUE

COMBINATION OF SUKEGAWA AND DYE IS IMPROPER

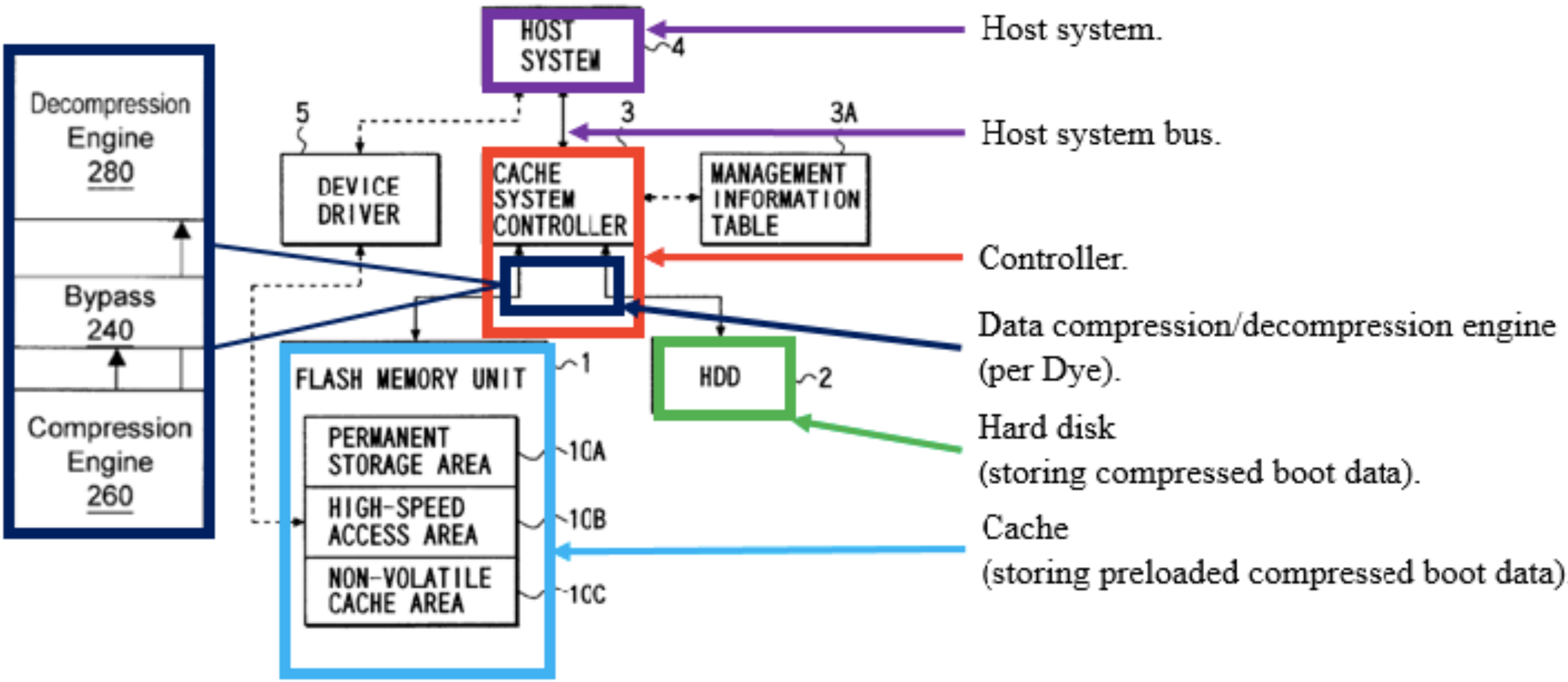
DYE DISCLOSES A COMPRESSION ENGINE IN A FLASH MEMORY CONTROLLER (‘1737, ‘1738 IPRS)

CEFMC MEANS “COMPRESSION ENHANCED FLASH MEMORY CONTROLLER “



Dye at Figure 3

SUKEGAWA AND DYE COMBINATION BASED ON IMPERMISSIBLE HINDSIGHT (‘1737, ‘1738 IPRS)



Sukegawa FIG. 1 and Dye FIG. 3 (combined excerpts, annotated).

‘1737 Petition at 16

SUKEGAWA AND DYE COMBINATION BASED ON IMPERMISSIBLE HINDSIGHT (‘1737, ‘1738 IPRS)

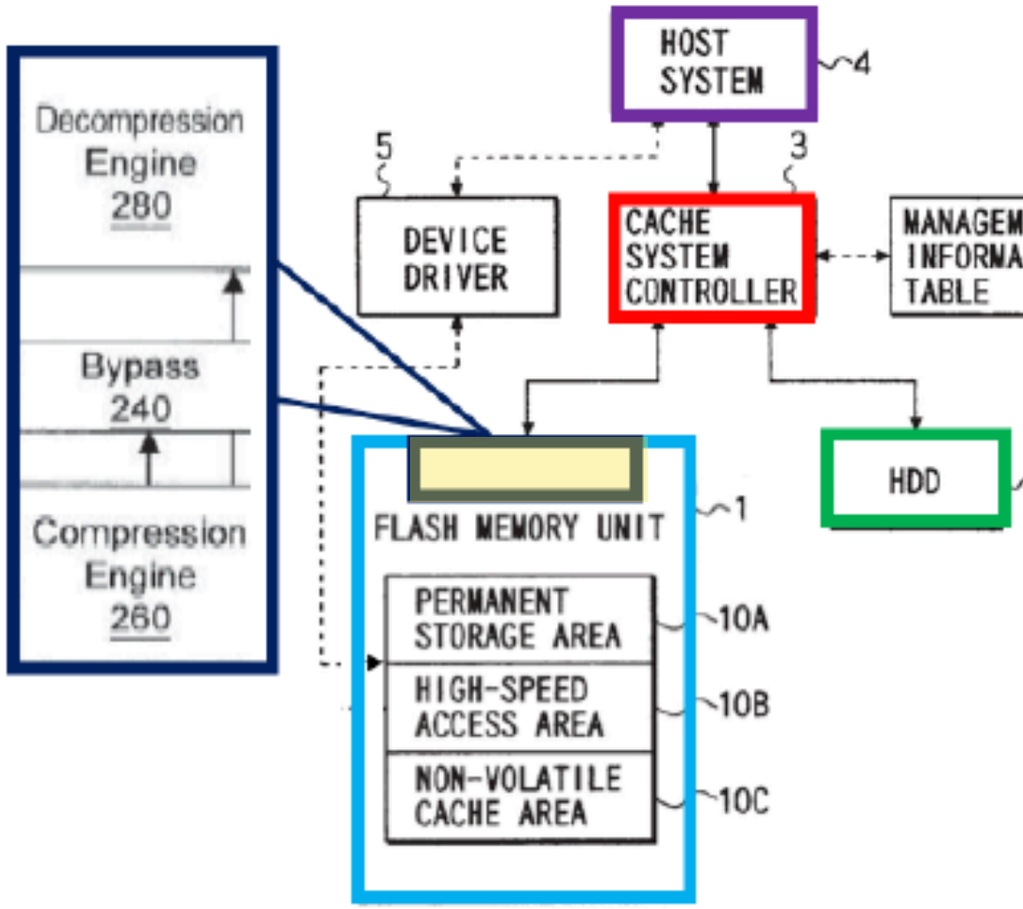
DR. BACK’S OPINION REGARDING THE COMBINATION OF SUKEGAWA AND DYE:

outgoing from flash memory.[92] To the extent a POSITA would have modified Sukegawa’s system in view of Dye, such a modification would have resulted in Dye’s data compression/decompression engine being embedded in Sukegawa’s flash memory unit:

Dr. Back ‘1737 Declaration (Ex. 2008) at ¶ 114

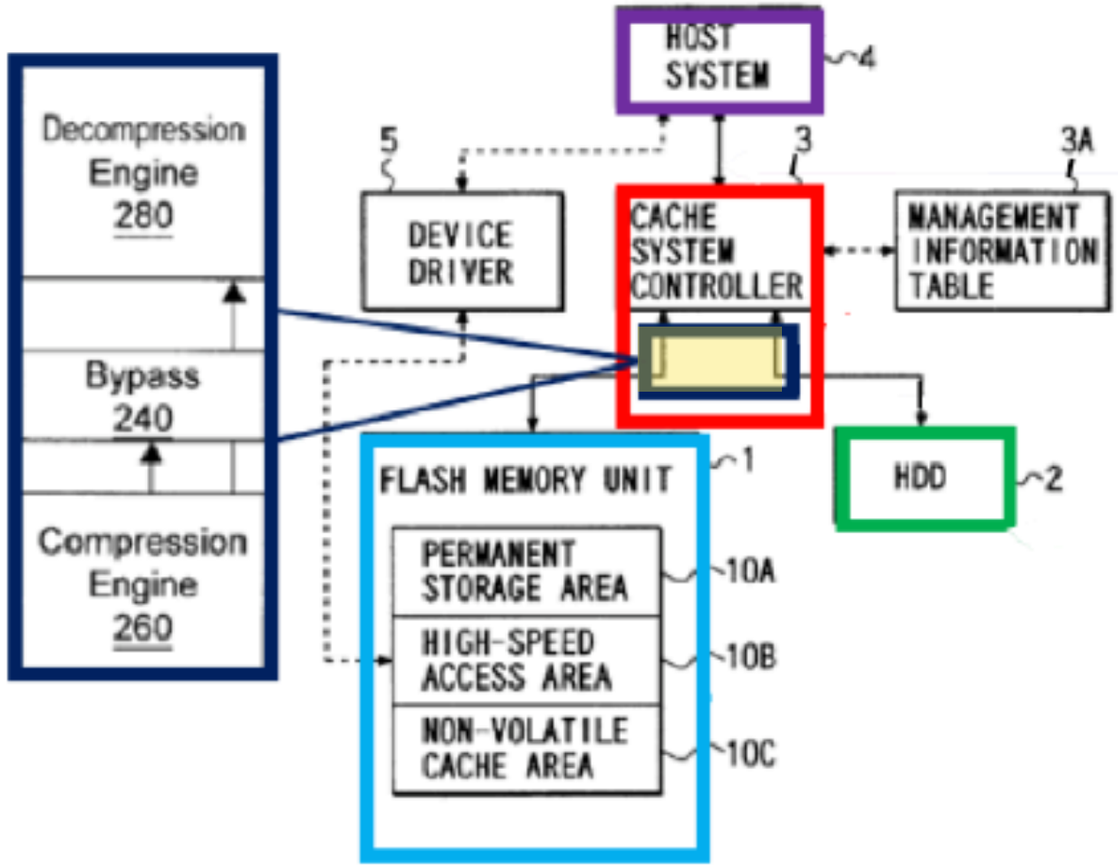
SUKEGAWA AND DYE COMBINATION BASED ON IMPERMISSIBLE HINDSIGHT (‘1737, ‘1738 IPRS)

DR. BACK’S OPINION REGARDING THE COMBINATION OF SUKEGAWA AND DYE:



Location of Dye’s Engine To The Extent Combination of Sukegawa and Dye is Proper

NOT



Excerpt of Apple’s Diagram (Petition, p. 16)

Dr. Back ‘1737 Declaration (Ex. 2008) at ¶ 114

SUKEGAWA AND DYE COMBINATION BASED ON IMPERMISSIBLE HINDSIGHT ('1737, '1738 IPRS)

memory unit 1 to controller 3.[93] However, Dr. Neuhauser does not explain why a POSITA would have chosen such a needlessly complex approach to modify Sukegawa's system versus the simpler approach actually taught in Dye—namely, compressing and decompressing data transferred into and out of a flash memory. Therefore, it appears that Dr. Neuhauser and Apple used the '862 Patent as a blueprint to reconstruct the claimed invention. I understand that a patent cannot be found obvious based on such a hindsight analysis.

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 115

POSITA WOULD NOT COMBINE SUKEGAWA AND DYE DUE TO COST AND COMPLEXITY ISSUES ('1737, '1738 IPRS)

AS REALTIME ARGUES IN ITS RESPONSES, DR. NEUHAUSER STATES DYE'S TECHNIQUE PRESENTS COMPLEXITY ISSUES:

THE WITNESS: This is another one of those examples of an incomplete hypothetical, because it depends on the cost of data compression. Data compression in Dye is not cheap.

Q In what sense?

A Well, so there's -- in Dye -- let's just speak about Dye, for example. That's the most concrete thing and the important issue here.

One of the objectives of Dye is to make data compression faster than transferring information without data compression, and to do that, Dye has to use, or recommends -- I don't know that he has to use this, but the preferred embodiment is to use a highly parallel data compression engine, which is quite complex, okay?

I don't think that that's a trivial cost. Remember we're talking about 2000. There was a lot of different technology a person could use, and it would depend on a lot of factors, like volume and so forth, but data compression is not without cost.

Dr. Neuhauser Transcript (Ex. 2026) at 97:4-23; see also '1737 Patent Owner Response at 58-59, '1738 Patent Owner Response at 61-62

POSITA WOULD NOT COMBINE SUKEGAWA AND DYE DUE TO COMPLEXITY ISSUES ('1737, '1738 IPRS)

AS REALTIME ARGUES IN ITS RESPONSES, DR. NEUHAUSER STATES DYE'S TECHNIQUE PRESENTS COMPLEXITY ISSUES:

Q What do you mean by "cost" when you say that?

A Well, there's a lot of different costs, but the cost I'm thinking about right now is the actual monetary cost of the system, right, because nonvolatile memory costs something, RAM costs something, the support logic costs something, and compression requires logic, specialized logic in the case of Dye, and that costs something.

So you'd have to know what all of those costs were before a statement like the one you just made, you could really assess that.

Dr. Neuhauser Transcript (Ex. 2026) at 97:24-98:10; see also '1737 Patent Owner Response at 58-59, '1738 Patent Owner Response at 61-62

POSITA WOULD NOT COMBINE SUKEGAWA AND DYE DUE TO COMPLEXITY ISSUES ('1737, '1738 IPRS)

AS REALTIME ARGUES IN ITS RESPONSES, DR. BACK STATES DYE'S TECHNIQUE PRESENTS COMPLEXITY ISSUES:

combination. The additional cost and complexity of adding Dye's compression to the system of Sukegawa would act as a significant disincentive for a POSA to make the proposed combination, and Dr. Neuhauser's declaration does not address those factors or explain why it would have been obvious for a POSA to make that combination in spite of them. A POSA is certainly sufficiently intelligent and experienced to know not to undertake a complex and costly modification simply because it may have some benefits in isolation, without evaluating whether those benefits warrant the undertaking given its costs and challenges.

Dr. Back '1737 Declaration (Ex. 2027) at ¶ 50; see also '1737 Patent Owner Response at 58-59, '1738 Patent Owner Response at 61-62

POSITA WOULD NOT COMBINE SUKEGAWA AND DYE DUE TO COMPLEXITY INCREASES ('1737, '1738 IPRS)

AS REALTIME ARGUES IN ITS RESPONSES, DR. BACK STATES DYE'S TECHNIQUE PRESENTS COMPLEXITY ISSUES:

46. I disagree with Dr. Neuhauser that a POSA would have been motivated to make such a combination. As I explain in further detail below, modifying Sukegawa to incorporate Dye's compression system would require substantial additional cost and would create significant engineering complexity. Dr. Neuhauser does not addresses those costs or complexities in his declaration, nor does he explain why a POSA would have been motivated to make the proposed combination despite those costs.

Dr. Back '1737 Declaration (Ex. 2027) at ¶ 46; see also '1737 Patent Owner Response at 58-59, '1738 Patent Owner Response at 61-62

BURROWS DISCOURAGES USE OF DYE'S COMPRESSION TECHNIQUE TO ACCELERATE BOOT-UP ('1737, '1738 IPRS)

DYE'S ONLY COMPRESSION TECHNIQUE USES LZ ENCODING. ('1737 PET. AT 49.)

BURROW SPECIFIES USING LZ COMPRESSION SLOWS BOOT-UP VERSUS USING NO COMPRESSION:

System	Time for phase (seconds)					Total
	MakeDir	Copy	ScanDir	ReadAll	Make	
unmodified LFS	1 ± 1	4 ± 1	4 ± 1	4 ± 1	58 ± 2	71 ± 2
no compression	0 ± 1	5 ± 1	4 ± 1	4 ± 1	64 ± 2	78 ± 1
Algorithm 1	1 ± 1	5 ± 1	3 ± 1	5 ± 1	63 ± 2	77 ± 2
Algorithm 2	1 ± 1	6 ± 1	4 ± 1	5 ± 1	67 ± 2	83 ± 2
LZRW1-A	1 ± 1	5 ± 1	3 ± 1	5 ± 1	65 ± 2	79 ± 1
LZRW3-A	0 ± 1	5 ± 1	3 ± 1	5 ± 1	66 ± 1	80 ± 1

This table shows the time in seconds for the Andrew file system benchmark running on six LFS configurations: unmodified LFS, and our modified LFS with five different compression algorithms. The values given are the mean of three runs, each on a newly rebooted and otherwise idle DECstation 5000/200 using an RZ55 disk.

Burrows at 20 (Table 3)

POSITA WOULD NOT INCORPORATE SETTSU'S COMPRESSION TECHNIQUE INTO SUKEGAWA ('1737, '1738 IPRS)

transferred to memory 2 during start-up.[98] A POSITA would not have understood how Settsu's teachings relate to Sukegawa's cooperative storage system in which copies of files are stored in both flash and hard disk memory nor to Dye's flash memory system with an embedded compression/decompression engine.

121. Moreover, Apple and Dr. Neuhauser do not explain how a POSITA would structure Sukegawa's storage system using compressed OS functional modules in view of Settsu's teachings.[99] Apple's and Dr. Neuhauser's proposed modification of Sukegawa's cooperative storage system to accommodate Settsu's teachings would require significant engineering effort and be non-obvious to a POSITA. For example, Sukegawa specifies that a data storage utility program

Dr. Back '1737 Declaration (Ex. 2008) at ¶¶ 120-21

POSITA WOULD NOT INCORPORATE ZWIEGINCEW'S COMPRESSION TECHNIQUE INTO SUKEGAWA ('1737, '1738 IPRS)

122. As described above in Section VII.B., Apple mischaracterizes Zwiegincew's teachings as being directed to slow boot times and page faults during the boot process. Rather, Zwiegincew describes techniques to reduce page faults for application programs during operation of the system after boot-up.[101] As also described in Section VII.B., a POSITA would not have found Zwiegincew's teachings relevant to the challenges addressed by the '862 claims nor the challenges described in Sukegawa and Dye. A POSITA, therefore, would not have been motivated by Zwiegincew to modify Sukegawa to stored compressed boot data in Sukegawa's HDD 2.

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 122

IPR2016-01737, -01738, -01739 ISSUE

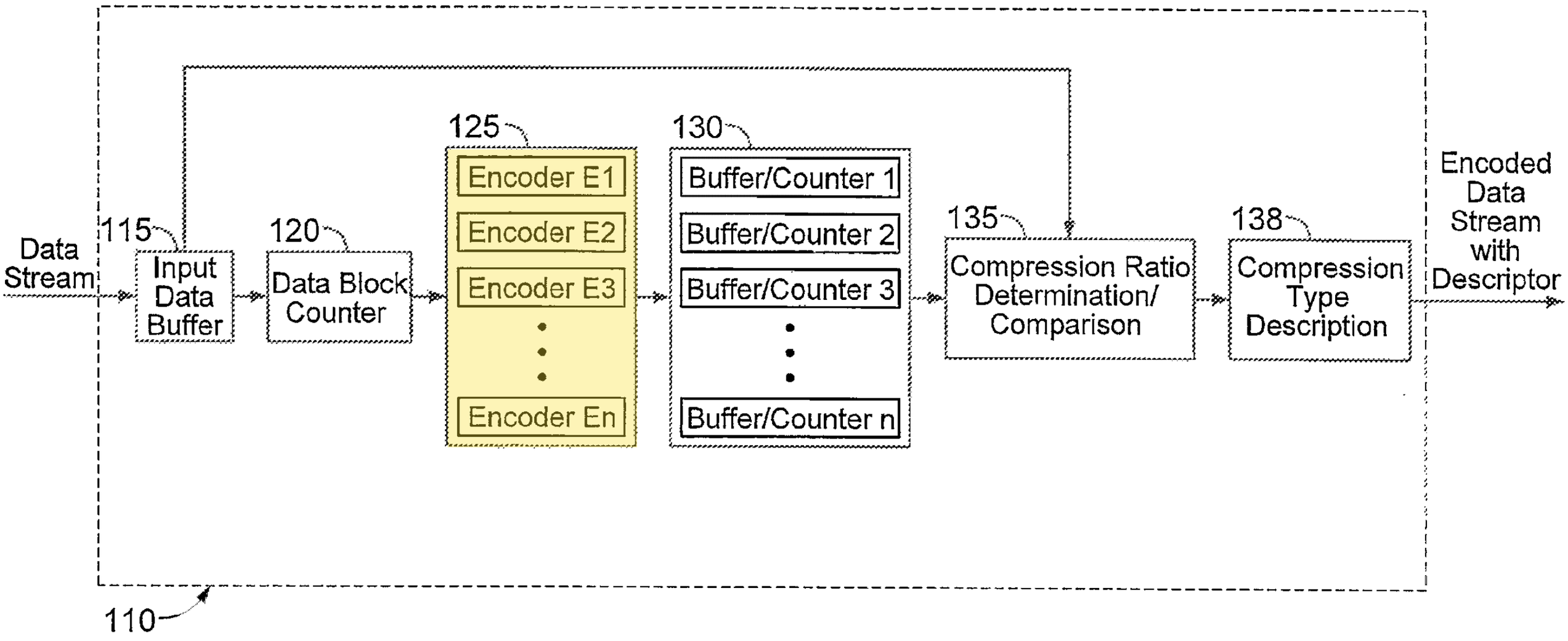
DYE DOES NOT RENDER OBVIOUS A “PLURALITY OF ENCODERS”

‘862 CLAIMS 34, 46, 58, 70, 82, 94 INCLUDE A PLURALITY OF ENCODERS TO ENCODE BOOT DATA (‘1737, ‘1738, ‘1739 IPRS)

34. The method of claim **1**, wherein a plurality of encoders was utilized to encode the portion of the boot data in the compressed form.

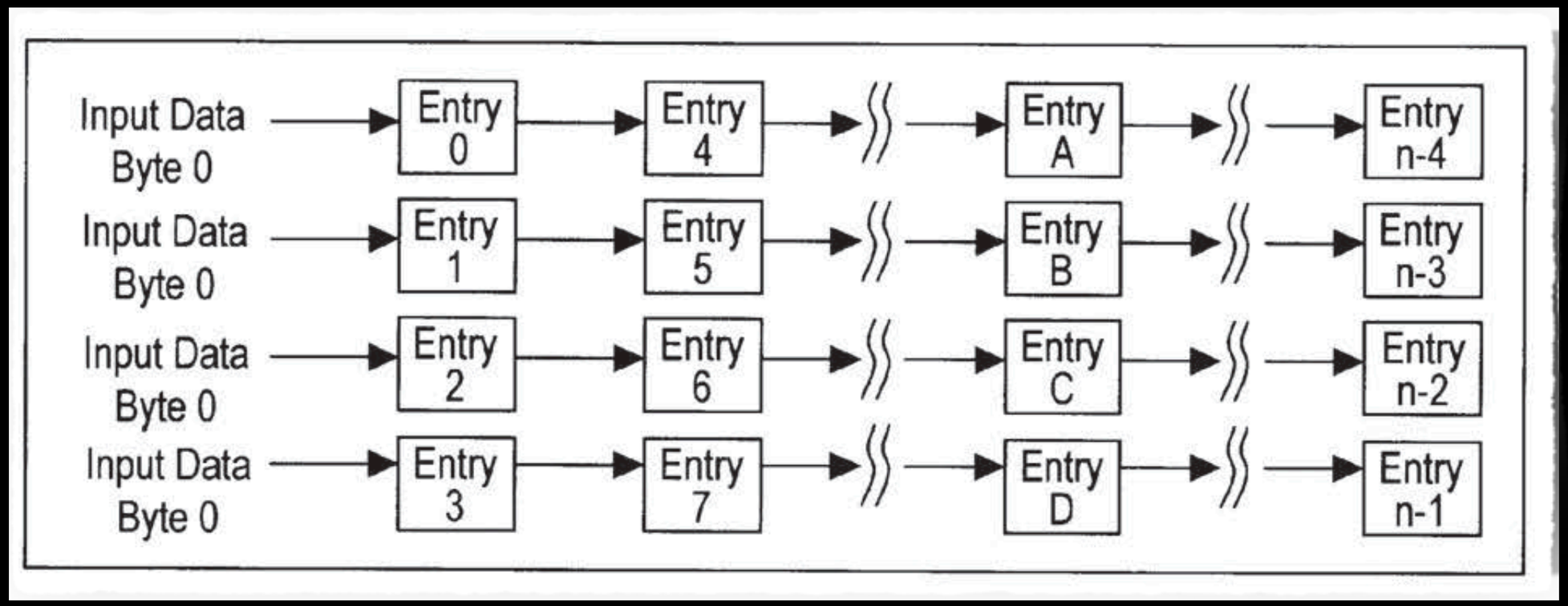
‘862 Patent at Claim 34

‘862 SPECIFICATION DISCLOSES A PLURALITY OF ENCODERS EACH ENCODING ITS OWN DATA STREAM



‘862 Patent at Figure 9

DYE DISCLOSES A SINGLE ENCODER THAT USES A "PARALLEL ALGORITHM" ('1737, '1738 IPRS)



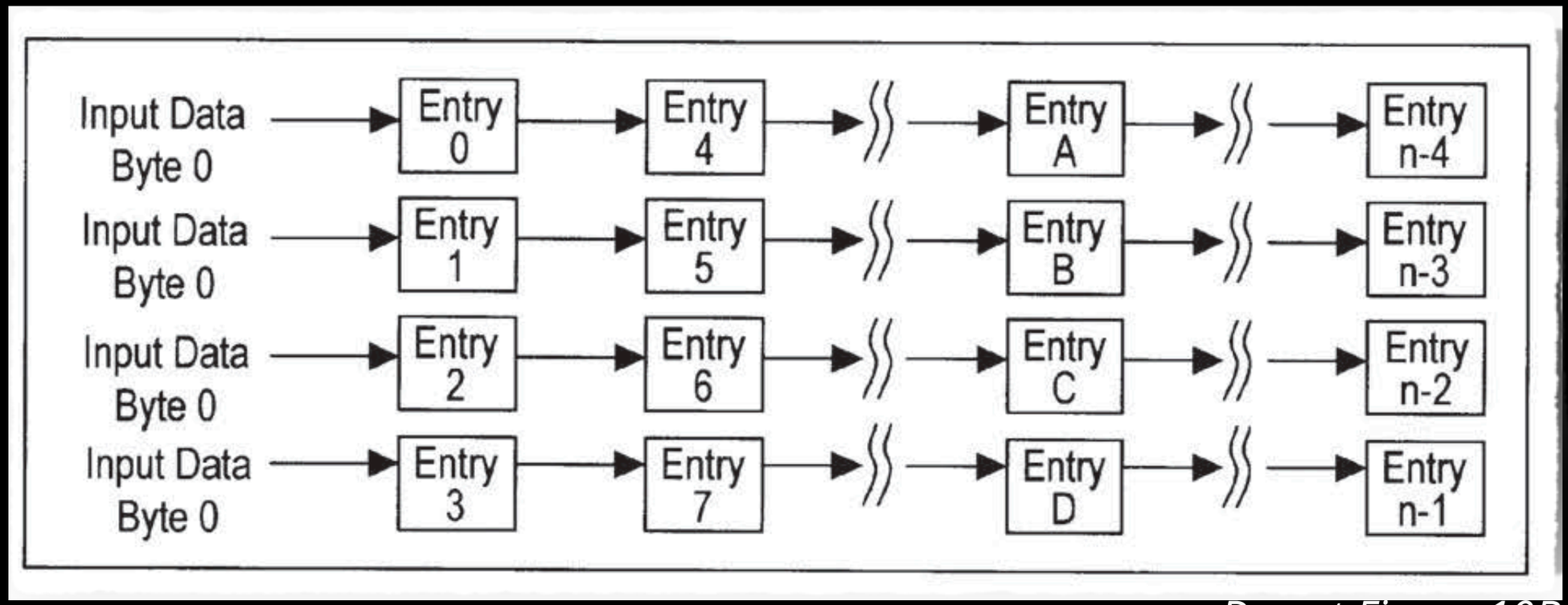
Dye at Figure 10B

FIG. **10**B—Parallel Algorithm

The preferred embodiment of the present invention provides a parallel implementation of dictionary based (or history table based) compression/decompression. By

Dye at 18:60-63

DYE DISCLOSES A SINGLE ENCODER THAT USES A "PARALLEL ALGORITHM" ('1737, '1738 IPRS)



Dye at Figure 10B

The parallel algorithm comprises paralleling three parts of the serial algorithm: the history table (or history window), analysis of symbols and compressed stream selection, and the output generation. In the preferred embodiment the data-flow through the history table becomes a 4 symbol parallel flow instead of a single symbol history table. Also, 4 symbols are analyzed in parallel, and multiple compressed

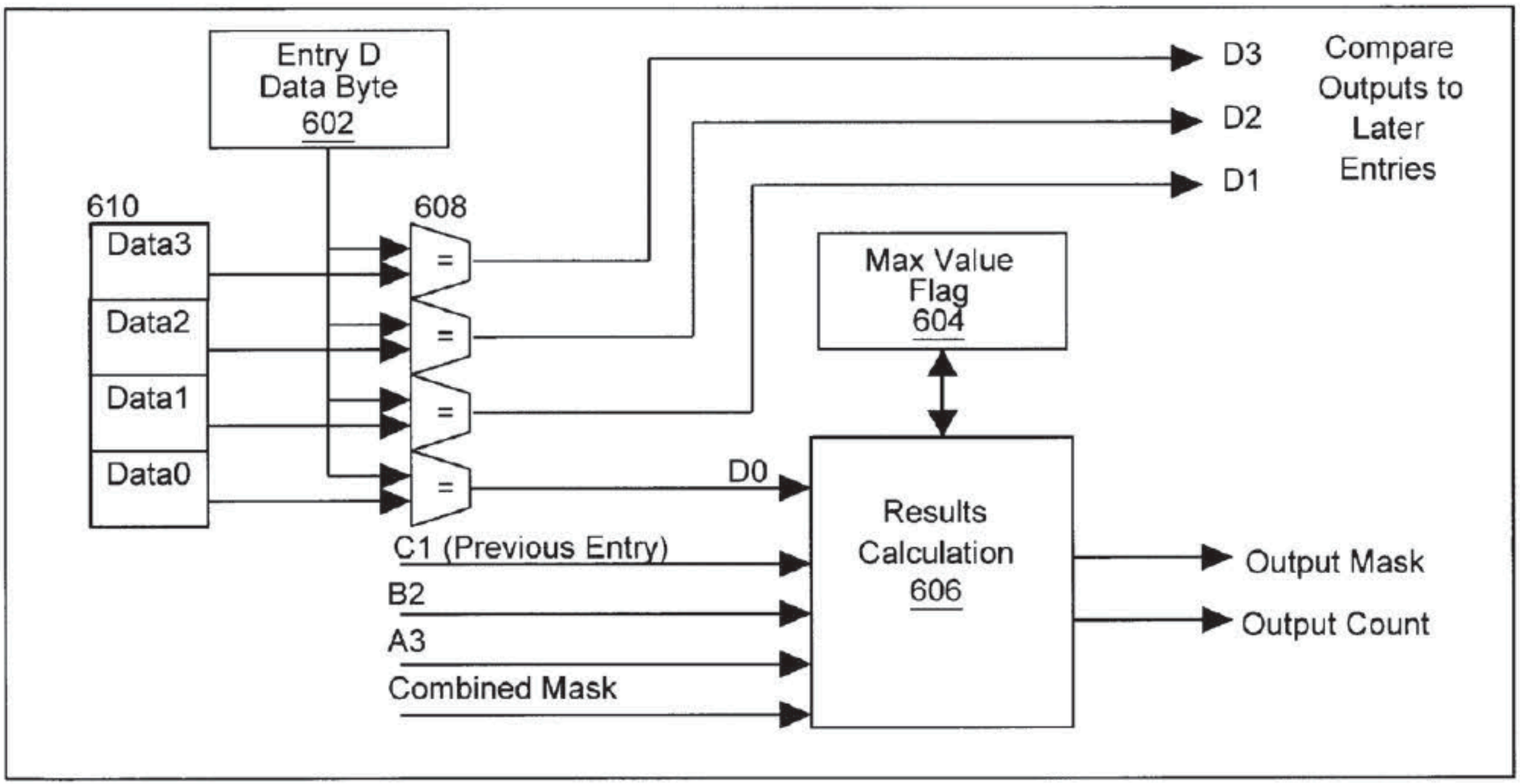
Dye at 19:12-17

POSITA WOULD UNDERSTAND DYE DISCLOSES A SINGLE ENCODER ('1737, '1738 IPRS)

130. For all five instituted grounds, Apple and Dr. Neuhauser rely on Dye as disclosing "a plurality of encoders," as recited in dependent claims 34, 58, and 94. Rather, Dye describes that the prior art approach, shown in Fig. 10B, encodes a single *algorithm*. Dye, on the other hand, Dye disclose a purported improvement of the prior art by using a single encoder that distributes the encoding calculations among several stages, as shown in Fig. 10A.[110] But each of these stages is not a separate encoder—on the contrary, each unit is a part of Dye's single encoder.[111]

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 130

DYE'S FIGURE 13 SHOWS MODIFIED ALGORITHM OF A SINGLE ENCODER ('1737, '1738 IPRS)



Dye at Figure 13

DYE'S FIGURE 15 IS AN EXAMPLE OF TABLE USED DURING COMPRESSION OF A SINGLE ENCODER ('1737, '1738 IPRS)

Input Matches				New Counter Value	Output Counter	Output Mask	Reset Value
D0	C1	B2	A3				
1	1	1	1	Saved+4	Saved +4	10000	0
1	1	1	0	0	Saved+3	10001	1
1	1	0	1	1	Saved+2	10010	2
1	1	0	0	0	Saved+2	10011	2
1	0	1	1	2	Saved+1	10100	3
1	0	1	0	0	Saved+1	10101	3
1	0	0	1	1	Saved+1	10110	3
1	0	0	0	0	Saved+1	10111	3
0	1	1	1	3	Saved	11000	4
0	1	1	0	0	Saved	01111	1
0	1	0	1	1	Saved	11010	4
0	1	0	0	0	Saved	11011	4
0	0	1	1	2	Saved	11100	4
0	0	1	0	0	Saved	11101	4
0	0	0	1	1	Saved	11110	4
0	0	0	0	0	Saved	11111	4

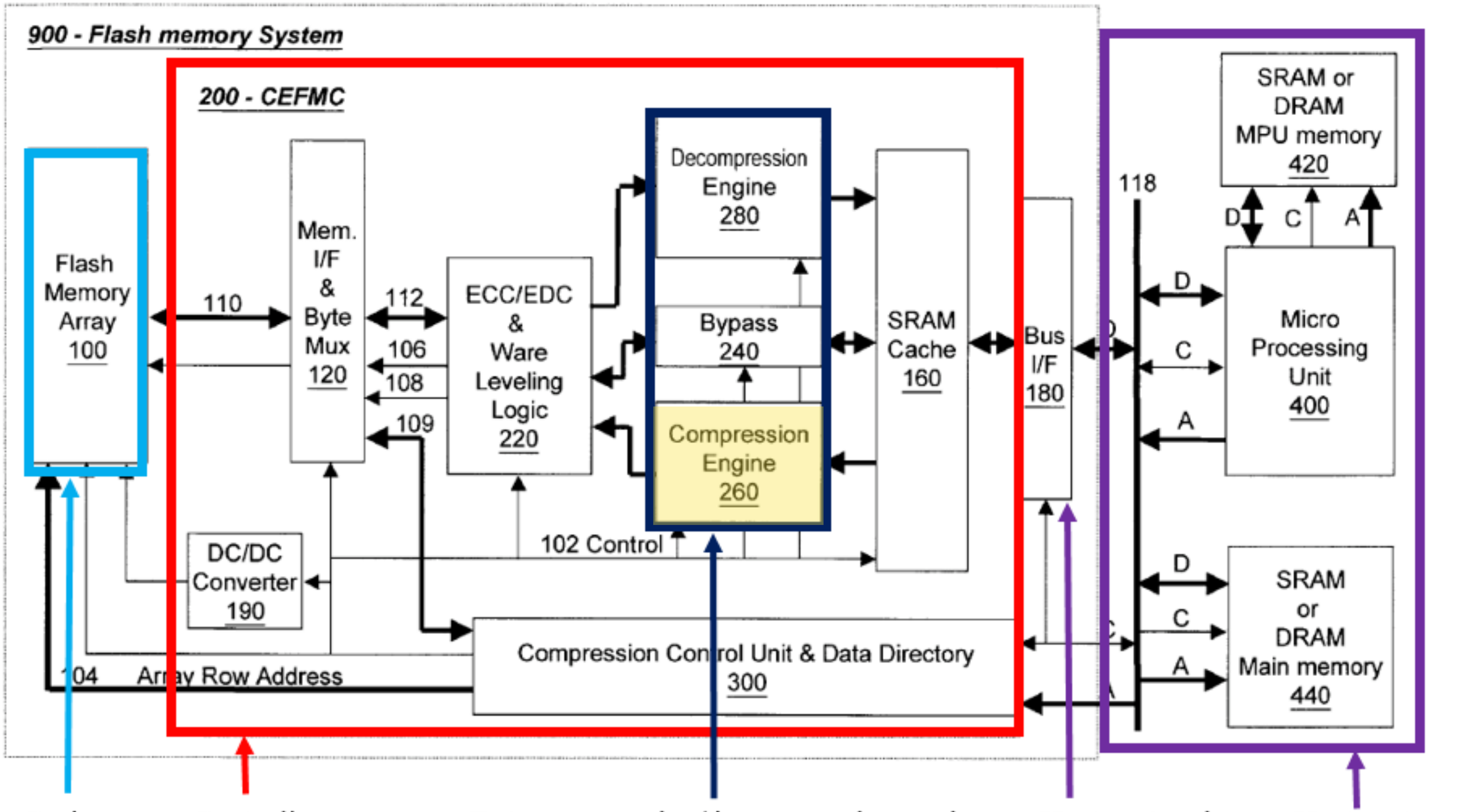
Dye at Figure 15

DYE'S FIGURES 13 AND 15 SHOW LOGIC OF A SINGLE ENCODER ('1737, '1738 IPRS)

132. Dye's description of Figures 13 and 15, however, does not support Dr. Neuhauser's opinion. Dye explains that Figure 13 illustrates Dye's modified algorithm, wherein the various components of the encoder logic work together, with each performing a subset of the encoding operations of Dye's single encoder.[113] And Figure 15 is an exemplary table used during the parallel compression that coordinates the parallel processes of the single encoder.

Dr. Back '1737 Declaration (Ex. 2008) at ¶ 132

APPLE'S PETITION SHOWS A SINGLE COMPRESSION ENGINE ('1737, '1738 IPRS)



Dye FIG. 3 (excerpt, annotated).

'1737 Petition at 14

APPLE'S "ENCODER" DEFINITION SHOWS THAT ENCODING COMPONENTS ARE NOT A PLURALITY OF ENCODERS

DICTIONARY SHOWS WINDOWS MEDIA ENCODER AND MP3 ENCODER ARE EXAMPLES OF "ENCODERS"—NOT INDIVIDUAL ENCODING COMPONENTS WITHIN AN ENCODER:

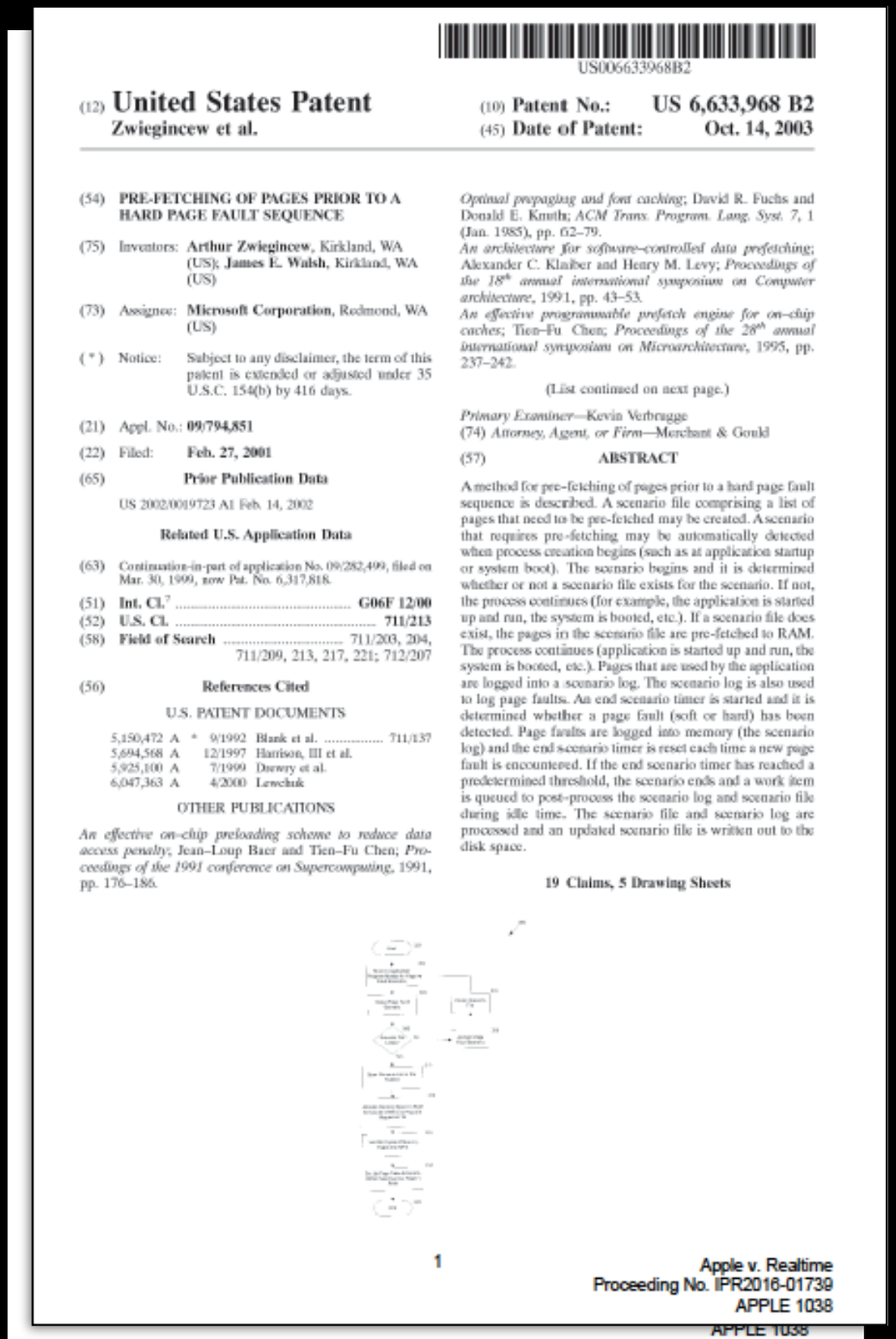
encoder *n.* **1.** In general, any hardware or software that encodes information—that is, converts the information to a particular form or format. For example, the Windows Media Encoder converts audio and video to a form that can be streamed to clients over a network. **2.** In reference to MP3 digital audio in particular, technology that converts a WAV audio file into an MP3 file. An MP3 encoder compresses a sound file to a much smaller size, about one-twelfth as large as the original, without a perceptible drop in quality. *Also called:* MP3 encoder. *See also* MP3, WAV. *Compare* rip, ripper.

Microsoft Computer Dictionary (Ex. 1041) at 4

IPR2016-01737, -01738, -01739 ISSUE

REALTIME'S MOTIONS TO EXCLUDE EVIDENCE

EX. 1038 SHOULD BE EXCLUDED AS HEARSAY AND IRRELEVANT ('1737, '1738, '1739 IPRS)



US006633968B2

(12) **United States Patent**
Zwiegincew et al.

(10) **Patent No.: US 6,633,968 B2**
(45) **Date of Patent: Oct. 14, 2003**

(54) **PRE-FETCHING OF PAGES PRIOR TO A HARD PAGE FAULT SEQUENCE**

(75) Inventors: **Arthur Zwiegincew**, Kirkland, WA (US); **James E. Walsh**, Kirkland, WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 416 days.

(21) Appl. No.: **09/794,851**

(22) Filed: **Feb. 27, 2001**

(65) **Prior Publication Data**

US 2002/0019723 A1 Feb. 14, 2002

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/282,499, filed on Mar. 30, 1999, now Pat. No. 6,317,818.

(51) **Int. Cl.**[7] **G06F 12/00**
(52) **U.S. Cl.** **711/213**
(58) **Field of Search** 711/203, 204, 711/209, 213, 217, 221; 712/207

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,150,472	A *	9/1992	Blank et al.	711/137
5,694,568	A	12/1997	Harrison, III et al.	
5,925,100	A	7/1999	Drewry et al.	
6,047,363	A	4/2000	Lewchuk	

OTHER PUBLICATIONS

An effective on-chip preloading scheme to reduce data access penalty; Jean–Loup Baer and Tien–Fu Chen; *Proceedings of the 1991 conference on Supercomputing*, 1991, pp. 176–186.

Optimal prepaging and font caching; David R. Fuchs and Donald E. Knuth; *ACM Trans. Program. Lang. Syst. 7*, 1 (Jan. 1985), pp. 62–79.

An architecture for software–controlled data prefetching; Alexander C. Klaiber and Henry M. Levy; *Proceedings of the 18th annual international symposium on Computer architecture*, 1991, pp. 43–53.

An effective programmable prefetch engine for on–chip caches; Tien–Fu Chen; *Proceedings of the 28th annual international symposium on Microarchitecture*, 1995, pp. 237–242.

(List continued on next page.)

Primary Examiner—Kevin Verbrugge
(74) *Attorney, Agent, or Firm*—Merchant & Gould

(57) **ABSTRACT**

A method for pre-fetching of pages prior to a hard page fault sequence is described. A scenario file comprising a list of pages that need to be pre-fetched may be created. A scenario that requires pre-fetching may be automatically detected when process creation begins (such as at application startup or system boot). The scenario begins and it is determined whether or not a scenario file exists for the scenario. If not, the process continues (for example, the application is started up and run, the system is booted, etc.). If a scenario file does exist, the pages in the scenario file are pre-fetched to RAM. The process continues (application is started up and run, the system is booted, etc.). Pages that are used by the application are logged into a scenario log. The scenario log is also used to log page faults. An end scenario timer is started and it is determined whether a page fault (soft or hard) has been detected. Page faults are logged into memory (the scenario log) and the end scenario timer is reset each time a new page fault is encountered. If the end scenario timer has reached a predetermined threshold, the scenario ends and a work item is queued to post-process the scenario log and scenario file during idle time. The scenario file and scenario log are processed and an updated scenario file is written out to the disk space.

19 Claims, 5 Drawing Sheets

1

Apple v. Realtime
Proceeding No. IPR2016-01739
APPLE 1038

APPLE 1038

Exhibit 1038 is a continuation-in-part of Exhibit 1010.

Exhibit 1038 should be excluded because Apple offers it to prove the truth of the matter being asserted regarding the disclosures of Zwiegincew (Exhibit 1010)—this constitutes impermissible hearsay per FRE 802 without an applicable exception.

Apple failed to establish that the information cited in Exhibit 1038 was publicly available and accessible prior to the '862 priority date.

Exhibit 1038 does not tend to make a fact of consequence more or less probable than it would be without this exhibit, thus it should be excluded per FRE 401, 402.

EXS. 1048-1049 SHOULD BE EXCLUDED AS LACKING AUTHENTICATION, HEARSAY, AND IRRELEVANT ('1737, '1738 IPRS)

Exhibits 1048 and 1049 appear to be websites that Apple failed to properly authenticate—thus each should be excluded per FRE 901, 902.

Exhibits 1048 and 1049 should be excluded because Apple offers each to prove the truth of the matter being asserted regarding the relative cost of RAM versus flash memory—this constitutes impermissible hearsay per FRE 802 without an applicable exception.

Dr. Neuhauser does not rely upon Exhibits 1048 and 1049 and each is published years after the '862 filing date. As such, the exhibits should be excluded as irrelevant per FRE 401, 402 because each does not tend to make a fact of consequence more or less probable than it would be without these exhibits.