

100404
 100404
 13281 U.S. PTO

Approved for use through 10/31/2002. OMB 0651-0032
 PTO/SB/05 (03-01)

UTILITY PATENT APPLICATION TRANSMITTAL <i>(Only for new nonprovisional applications under 37 CFR 1.53(b))</i>	Attorney Docket No. 111325-235000	
	First Inventor Mai NGUYEN, et al.	
	Title	SYSTEM AND METHOD FOR RIGHTS OFFERING AND GRANTING USING SHARED STATE VARIABLES
	Express Mail Label No.	

APPLICATION ELEMENTS <i>See MPEP chapter 600 concerning utility patent application contents.</i>	ADDRESS TO: Commissioner for Patents Box Patent Application Washington, DC 20231
--	--

1. <input checked="" type="checkbox"/> Fee Transmittal Form (e.g., PTO/SB/17) <i>(Submit an original and a duplicate for fee processing)</i> 2. <input type="checkbox"/> Applicant claims small entity status. See 37 CFR 1.27. 3. <input checked="" type="checkbox"/> Specification [Total Pages 43] <i>(preferred arrangement set forth below)</i> - Descriptive title of the invention - Cross Reference to Related Applications <i>(if applicable)</i> - Statement Regarding Fed sponsored R & D <i>(if applicable)</i> - Reference to sequence listing, a table, or a computer program listing appendix <i>(if applicable)</i> - Background of the Invention - Brief Summary of the Invention - Brief Description of the Drawings <i>(if filed)</i> - Detailed Description - Claim(s) - Abstract of the Disclosure 4. <input checked="" type="checkbox"/> Drawing(s) (35 U.S.C. 113) 23 Figures [Total Sheets 17] 5. Oath or Declaration [Total Pages <input type="checkbox"/>] a. <input type="checkbox"/> Newly executed (original or copy) b. <input checked="" type="checkbox"/> Copy from a prior application (37 CFR 1.63(d)) <i>(for continuation/divisional with Box 18 completed)</i> i. <input type="checkbox"/> DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b) 6. <input checked="" type="checkbox"/> Application Data Sheet. See 37 CFR 1.76	7. <input type="checkbox"/> CD-ROM or CD-R in duplicate, large table or Computer Program (Appendix) 8. Nucleotide and/or Amino Acid Sequence Submission <i>(if applicable, all necessary)</i> a. <input type="checkbox"/> Computer Readable Form (CRF) b. Specification Sequence Listing on: i. <input type="checkbox"/> CD-ROM or CD-R (2 copies; or ii. <input type="checkbox"/> paper c. <input type="checkbox"/> Statements verifying identity of above copies
--	--

ACCOMPANYING APPLICATION PARTS
9. <input type="checkbox"/> Assignment Papers (cover sheet & document(s)) 10. <input type="checkbox"/> 37 CFR 3.73(b) Statement <input type="checkbox"/> Power of Attorney <i>(when there is an assignee)</i> 11. <input type="checkbox"/> English Translation Document <i>(if applicable)</i> 12. <input type="checkbox"/> Information Disclosure Statement (IDS)/PTO-1449 <input type="checkbox"/> Copies of IDS Citations 13. <input type="checkbox"/> Preliminary Amendment 14. <input checked="" type="checkbox"/> Return Receipt Postcard (MPEP 503) <i>(Should be specifically itemized)</i> 15. <input type="checkbox"/> Certified Copy of Priority Document(s) <i>(if foreign priority is claimed)</i> 16. <input type="checkbox"/> Nonpublication request under 35 U.S.C. 122(b)(2)(B)(i). Applicant must attach form PTO/SB/35 or its equivalent. 17. <input type="checkbox"/> Other: _____

18. If a CONTINUING APPLICATION, check appropriate box, and supply the requisite information below and in a preliminary amendment, or in an Application Data Sheet under 37 CFR 1.76:

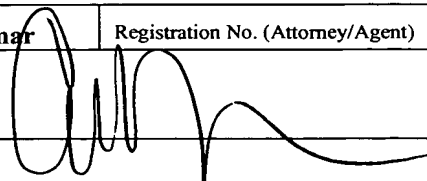
Continuation Continuation-in-part (CIP) of prior application No: 10/162,212

Prior application information: Examiner Not Yet Assigned Group / Art Unit: 3621

For CONTINUATION OR DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 5b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

19. CORRESPONDENCE ADDRESS	
<input checked="" type="checkbox"/> Customer Number or Bar Code Label	22204

Name			
Address			
City	State	Zip Code	
Country	Telephone	Fax	

Name (Print/Type)	Carlos R. Villamar	Registration No. (Attorney/Agent)	43,224
Signature		Date	October 4, 2004

100404

3281 U.S. Patent

FEE TRANSMITTAL FOR FY 2004

Patent fees are subject to annual revision.

Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT **\$1,132.00**

Complete if Known

Application Number	Not Yet Assigned
Filing Date	October 4, 2004
First Named Inventor	Mai NGUYEN, et al.
Examiner Name	Not Yet Assigned
Art Unit	Not Yet Assigned
Attorney Docket No.	111325-235000

METHOD OF PAYMENT (check all that apply)

Check Credit Card Money Order Other None

Deposit Account:

Deposit Account Number **19-2380**

Deposit Account Name **Nixon Peabody LLP**

The Commissioner is authorized to: (check all that apply)

Charge fee(s) indicated below Credit any overpayments

Charge any additional fee(s)

Charge fee(s) indicated below, except for the filing fee to the above-identified deposit account.

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity		Small Entity		Fee Description
Fee Code	Fee (\$)	Fee Code	Fee (\$)	
1051	130	2051	65	Surcharge - late filing fee or oath
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet
1053	130	1053	130	Non-English specification
1812	2,520	1812	2,520	For filing a request for <i>ex parte</i> reexamination
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action
1251	110	2251	55	Extension for reply within first month
1252	430	2252	215	Extension for reply within second month
1253	980	2253	490	Extension for reply within third month
1254	1,530	2254	765	Extension for reply within fourth month
1255	2,080	2255	1,040	Extension for reply within fifth month
1401	340	2401	170	Notice of Appeal
1402	340	2402	170	Filing a brief in support of an appeal
1403	340	2403	150	Request for oral hearing
1451	1,510	1451	1,510	Petition to institute a public use proceeding
1452	110	2452	55	Petition to revive - unavoidable
1453	1,370	2453	685	Petition to revive - unintentional
1501	1,370	2501	685	Utility issue fee (or reissue)
1502	490	2502	245	Design issue fee
1503	660	2503	330	Plant issue fee
1460	130	1460	130	Petitions to the Commissioner
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)
1806	180	1806	180	Submission of Information Disclosure Stmt
8021	40	8021	40	Recording each patent assignment per property (times number of properties)
1809	790	2809	395	Filing a submission after final rejection (37 CFR 1.129(a))
1810	790	2810	395	For each additional invention to be examined (37 CFR 1.129(b))
1801	790	2801	395	Request for Continued Examination (RCE)
1802	900	1802	900	Request for expedited examination of a design application

Other fee (specify) _____

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) **(\$ 0)**

CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]

I hereby certify that this correspondence is being:

- deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop _____, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450
- transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) _____.

Date _____

Signature _____

Typed or printed name _____

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee (\$)	Small Entity Fee Code	Small Entity Fee (\$)	Fee Description	Fee Paid
1001	790	2001	395	Utility filing fee	790.00
1002	350	2002	175	Design filing fee	
1003	550	2003	275	Plant filing fee	
1004	790	2004	395	Reissue filing fee	
1005	160	2005	80	Provisional filing fee	

SUBTOTAL (1) **\$790.00**

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

Total Claims	Extra Claims	Fee from below	Fee Paid
39	-20** = 19	X 18.00	= \$342.00
Independent Claims	3	-3** =	X = 0
Multiple Dependent		X	= 0

Large Entity Fee Code	Large Entity Fee (\$)	Small Entity Fee Code	Small Entity Fee (\$)	Fee Description
1202	18	2202	9	Claims in excess of 20
1201	88	2201	44	Independent claims in excess of 3
1203	300	2203	150	Multiple dependent claim, if not paid
1204	88	2204	44	** Reissue independent claims over original patent
1205	18	2205	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) **\$342.00**

**or number previously paid, if greater; For Reissues, see above

SUBMITTED BY

Name (Print/Type)	Carlos R. Villamar	Registration No. (Attorney/Agent)	43,224	Telephone	(202) 585-8204
Signature		Date	October 4, 2004		

Complete (if applicable)

SEND TO: Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

**SYSTEM AND METHOD FOR RIGHTS OFFERING AND GRANTING
USING SHARED STATE VARIABLES**

RELATED APPLICATION DATA

[0001] This application is a continuation-in-part application of co-pending application Ser. No. 10/162,212 filed on June 5, 2002, which is a continuation-in-part application of application Ser. No. 09/867,745 filed on May 31, 2001, and which claims benefit from U.S. provisional application Ser. No. 60/296,113, filed in June 7, 2001, U.S. provisional application, Serial No. 60/331,625 filed in November 20, 2001, and U.S. provisional application Ser. No. 60/331,624 filed on November 20, 2001, the entire disclosures of all of which are hereby incorporated by reference herein.

FIELD OF THE INVENTION

[0002] The present invention generally relates to offering and granting of rights and more particularly to a method, system and device for offering and granting of rights using shared state variables.

BACKGROUND OF THE INVENTION

[0003] The digital age has greatly increased concerns about ownership, access, and control of copyrighted information, restricted services and valuable resources. Rapid evolution and wide deployment has occurred for computers, and other electronic devices such as cellular phones, pagers, PDAs, and e-book readers, and these devices are interconnected through communication links including the Internet, intranets and other networks. These interconnected devices are especially conducive to publication of content, offering of services and availability of resources electronically.

[0004] One of the most important issues impeding the widespread distribution of digital works (i.e. documents or other content in forms readable by computers), via electronic means, and the Internet in particular, is the current lack of ability to enforce the intellectual property rights of content

owners during the distribution and use of digital works. Efforts to resolve this problem have been termed "Intellectual Property Rights Management" ("IPRM"), "Digital Property Rights Management" ("DPRM"), "Intellectual Property Management" ("IPM"), "Rights Management" ("RM"), and "Electronic Copyright Management" ("ECM"), collectively referred to as "Digital Rights Management (DRM)" herein. There are a number of issues to be considered in effecting a DRM System. For example, authentication, authorization, accounting, payment and financial clearing, rights specification, rights verification, rights enforcement, and document protection issues should be addressed. U.S. patents 5,530,235, 5,634,012, 5,715,403, 5,638,443, and 5,629,980, the disclosures of which are incorporated herein by reference, disclose DRM systems addressing these issues.

[0005] Two basic DRM schemes have been employed, secure containers and trusted systems. A "secure container" (or simply an encrypted document) offers a way to keep document contents encrypted until a set of authorization conditions are met and some copyright terms are honored (e.g., payment for use). After the various conditions and terms are verified with the document provider, the document is released to the user in clear form. Commercial products such as Cryptolopes and Digiboxes fall into this category. Clearly, the secure container approach provides a solution to protecting the document during delivery over insecure channels, but does not provide any mechanism to prevent legitimate users from obtaining the clear document and then using and redistributing it in violation of content owners' intellectual property.

[0006] In the "trusted system" approach, the entire system is responsible for preventing unauthorized use and distribution of the document. Building a trusted system usually entails introducing new hardware such as a secure processor, secure storage and secure rendering devices. This also requires that all software applications that run on trusted systems be certified to be trusted. While building tamper-proof trusted systems is a real challenge to

existing technologies, current market trends suggest that open and untrusted systems, such as PC's and workstations using browsers to access the Web, will be the dominant systems used to access digital works. In this sense, existing computing environments such as PC' s and workstations equipped with popular operating systems (e.g., Windows, Linux, and UNIX) and rendering applications, such as browsers, are not trusted systems and cannot be made trusted without significantly altering their architectures. Of course, alteration of the architecture defeats a primary purpose of the Web, i.e. flexibility and compatibility.

[0007] Some DRM systems allow content owners to specify usage rights and conditions, and associate them with content. These usage rights control how the recipient thereof can use the content. Usually after a content distributor or consumer has completed selecting and ordering specific content, the content is delivered either electronically from some content repository or via a conventional distribution channel to the recipient, such as tangible media sent via a common carrier. Corresponding DRM systems used by the recipient, for example the distributor or consumer, will then interpret the rights and conditions associated with the content, and use them to control how the content is distributed and/or used. Examples of usage rights include view, print and extract the content, and distribute, repackage and loan content. Associated conditions may include any term upon which the rights may be contingent such as payment, identification, time period, or the like.

[0008] U.S. patent 5,634,012, discloses a system for controlling the distribution of digital documents. Each rendering device has a repository associated therewith. A predetermined set of usage transaction steps define a protocol used by the repositories for enforcing usage rights associated with a document. Usage rights persist with the document content. The usage rights can permit various manners of use such as, viewing only, use once,

distribution, and the like. Usage rights can be contingent on payment or other conditions.

[0009] However, there are limitations associated with the above-mentioned paradigms wherein only usage rights and conditions associated with content are specified by content owners or other grantors of rights. Once purchased by an end user, a consumer, or a distributor, of content along with its associated usage rights and conditions has no means to be legally passed on to a next recipient in a distribution chain. Further the associated usage rights have no provision for specifying rights to derive other rights, i.e. rights to modify, transfer, offer, grant, obtain, transfer, delegate, track, surrender, exchange, transport, exercise, revoke, or the like. Common content distribution models often include a multi-tier distribution and usage chain. Known DRM systems do not facilitate the ability to prescribe rights and conditions for all participants along a content distribution and usage chain. Therefore, it is difficult for a content owner to commercially exploit content unless the owner has a relationship with each party in the distribution chain.

SUMMARY OF THE INVENTION

[0010] Exemplary aspects of the present invention include a method, system and device for sharing rights adapted to be associated with items, the method and system including generating at least one of usage rights and meta-rights for the items; defining, via the usage rights, a manner of use for the items; and defining, via the meta-rights, a manner of rights transfer for the items. The device including receiving at least one of usage rights and meta-rights for the items; interpreting, via the usage rights, a manner of use for the items; and interpreting, via the meta-rights, a manner of rights transfer for the items. The usage rights or the meta-rights include at least one state variable that is shared by one or more rights.

[0011] Still other aspects, features, and advantages of the present invention are readily apparent from the following detailed description, simply

by illustrating a number of exemplary embodiments and implementations, including the best mode contemplated for carrying out the present invention. The present invention is also capable of other and different embodiments, and its several details can be modified in various respects, all without departing from the spirit and scope of the present invention. Accordingly, the drawings and descriptions are to be regarded as illustrative in nature, and not as restrictive.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] Exemplary embodiments of this invention will be described in detail, with reference to the attached drawings in which:

[0013] Fig. 1 is a schematic diagram of a three-tier model for content distribution;

[0014] Fig. 2 is a schematic diagram illustrating rights offering and granting processes in the model of Fig. 1;

[0015] Fig. 3(a) is a schematic diagram of a simple supplier-consumer push model for rights generating, issuing and exercising;

[0016] Fig. 3(b) is a schematic diagram of a simple supplier-consumer pull model for rights generating, issuing and exercising;

[0017] Fig. 4 is a block diagram of a rights offering-granting architecture in accordance with the preferred embodiment;

[0018] Figs. 5a and 5b are workflow diagrams for examples of offering and granting rights between a rights supplier and a rights consumer with a push and pull model respectively;

[0019] Fig. 6 is a flow chart of a rights offer generation process in accordance with the preferred embodiment;

[0020] Fig. 7 is a flow chart of a rights offer consideration process in accordance with the preferred embodiment;

[0021] Fig. 8 is a flow chart of a rights offer customization process in accordance with the preferred embodiment;

[0022] Fig. 9 is block diagram of a DRM system that may be utilized in connection with the preferred embodiment;

[0023] Fig. 10 is a block diagram of an exemplary structure of a license containing usage rights and meta-rights of the preferred embodiment;

[0024] Fig. 11 is a schematic illustration of a rights label of the preferred embodiment;

[0025] Fig. 12 illustrates an exemplary system including a state-of-rights server;

[0026] Fig. 13 illustrates employing of a state variable in deriving exclusive usage rights;

[0027] Fig. 14 illustrates employing of a state variable in deriving inherited usage rights;

[0028] Fig. 15 illustrates employing of a state variable in deriving rights that are shared among a known set of rights recipients;

[0029] Fig. 16 illustrates employing of a state variable in deriving rights that are shared among a dynamic set of rights recipients;

[0030] Fig. 17 illustrates employing of a state variable in maintaining a state shared by multiple rights;

[0031] Fig. 18 illustrates employing of multiple state variables to represent one state of rights;

[0032] Fig. 19 illustrates a case where not all rights are associated with states;

[0033] Fig. 20 illustrates a case where not all rights which are associated with states are shared or inherited; and

[0034] Fig. 21 illustrates a case of rights sharing based on an offer which does not explicitly include meta-rights.

DETAILED DESCRIPTION

[0035] Prior to providing detailed description of the apparatus and method for offering and granting rights, a description of a DRM system that can be utilized to specify and enforce usage rights and meta-rights for specific content, services, or other items is first described below.

[0036] Fig. 9 illustrates DRM System 10 that includes a user activation component, in the form of activation server 20, that issues public and private key pairs, or other identification mechanisms, to content users in a protected fashion, as is well known. Typically, when a user uses DRM system 10 for the first time, the user installs software that works with, or includes, a rendering application for a particular content format. The software is installed in client environment 30, a computer associated with the content recipient, for example. The software is part of DRM 10 system and is used to enforce usage rights for protected content. During the activation process, some information is exchanged between activation server 20 and client environment 30. Client component 60 preferably is tamper resistant and contains the set of public and private keys issued by activation server 20 as well as other components, such as rendering components for example.

[0037] Rights label 40 is associated with content 42 and specifies usage rights and meta-rights that are available to a recipient, i.e. a consumer of rights, when corresponding conditions are satisfied. License Server 50 manages the encryption keys and issues licenses 52 for protected content 42. Licenses 52 embody the actual granting of rights, including usage rights and meta-rights, to an end user. For example, rights offer 40 may permit a user to view content for a fee of five dollars and print content for a fee of ten dollars, or it may permit a user to offer rights to another user, for example, by utilizing the concept of meta-rights described below. License 52 can be issued for the view right when the five dollar fee has been paid. Client component 60 interprets and enforces the rights, including usage rights and meta-rights, that

have been specified in the license. Rights label 40 and license 52 are described in detail below.

[0038] Fig. 11 illustrates rights label 40 in accordance with the preferred embodiment. Rights label 40 includes plural rights options 44. Each rights option 44 includes usage rights 44a, conditions 44b, and content specification 44c. Content specification 44c can include any mechanism for referencing, calling, locating, or otherwise specifying content 42 associated with rights offer 44.

[0039] As shown in Fig. 10, license 52 includes license 52a, grant 52b, and digital signature 52c. Grant 52b includes granted usage rights and/or meta-rights selected from label. The structure of the grant also includes one or more principals, to whom the specified usage rights and/or meta-rights are granted, a list of conditions, and state variables required to enforce the license. Like usage rights, access and exercise of the granted meta-rights are controlled by the condition list and state variables as described below.

[0040] Clear (unprotected) content can be prepared with document preparation application 72 installed on computer 70 associated with a content publisher, a content distributor, a content service provider, or any other party. Preparation of content consists of specifying the usage rights, meta-rights, and conditions under which content 42 can be used and distributed, associating rights label 40 with content 42 and protecting content 42 with some crypto algorithm. A rights language such as XrML can be used to specify the rights and conditions. However, the usage rights and meta-rights can be specified in any manner. Also, the rights can be in the form of a pre-defined specification or template that is merely associated with the content. Accordingly, the process of specifying rights refers to any process for associating rights with content. Rights label 40 associated with content 42 and the encryption key used to encrypt the content can be transmitted to license server 50.

[0041] Rights can specify transfer rights, such as distribution rights, and can permit granting of rights to others or the derivation of rights. Such rights are referred to as “meta-rights”. Meta-rights are the rights that one has to manipulate, modify, or otherwise derive other meta-rights or usage rights. Meta-rights can be thought of as usage rights to usage rights. Meta-rights can include rights to offer, grant, obtain, transfer, delegate, track, surrender, exchange, and revoke usage rights to/from others. Meta-rights can include the rights to modify any of the conditions associated with other rights. For example, a meta-right may be the right to extend or reduce the scope of a particular right. A meta-right may also be the right to extend or reduce the validation period of a right.

[0042] Often, conditions must be satisfied in order to exercise the manner of use in a specified right. For, example a condition may be the payment of a fee, submission of personal data, or any other requirement desired before permitting exercise of a manner of use. Conditions can also be “access conditions” for example, access conditions can apply to a particular group of users, say students in a university, or members of a book club. In other words, the condition is that the user is a particular person or member of a particular group. Rights and conditions can exist as separate entities or can be combined.

[0043] State variables track potentially dynamic states conditions. State variables are variables having values that represent status of an item, usage rights, license or other dynamic conditions. State variables can be tracked, by clearinghouse 90 license or server 30 another device, based on identification mechanisms in license 52. Further, the value of state variables can be used in a condition. For example, a usage right can be the right to print content 42 three times. Each time the usage right is exercised, the value of the state variable “number of prints” is incremented. In this example, when the value of the state variable is three, the condition is not longer satisfied and content 42 cannot be printed. Another example of a state variable is time. A

condition of license 52 may require that content 42 is printed within thirty days. A state variable can be used to track the expiration of thirty days. Further, the state of a right can be tracked as a collection of state variables. The collection of the change is the state of a usage right represents the usage history of that right.

[0044] A typical workflow for DRM system 10 is described below. A recipient, such as a user, operating within client environment 30 is activated for receiving content by activation server 20. This results in a public-private key pair (and some user/machine specific information) being downloaded to client environment 30 in the form of client software component 60 in a known manner. This activation process can be accomplished at any time prior to the issuing of a license.

[0045] When a user wishes to use protected content 42, the user makes a request for the content 42. For example, a user might browse a Web site running on Web server 80 associated with a grantor of rights such as a content distributor, using a browser installed in client environment 30, and attempt to download protected content 42. During this process, the user may go through a series of steps possibly including a fee transaction (as in the sale of content) or other transactions (such as collection of information). When the appropriate conditions and other prerequisites, such as the collection of a fee and verification that the user has been activated, are satisfied, Web server 80 contacts license server 50 through a secure communications channel, such as a channel using a Secure Sockets Layer (SSL). License server 50 then generates license 52 for the content and Web server 80 causes both protected content 42 and license 52 to be downloaded. License 52 can be downloaded from license server 50 or an associated device. Content 42 can be downloaded from computer 70 associated with a publisher, distributor, or other party.

[0046] Client component 60 in client environment 30 will then proceed to interpret license 52 and allow use of content 42 based on the rights and

conditions specified in license 52. The interpretation and enforcement of usage rights are well known generally. The steps above may take place sequentially or approximately simultaneously or in various order.

[0047] DRM system 10 addresses security aspects of protecting content 42. In particular, DRM system 10 may authenticate license 52 that has been issued by license server 50. One way to accomplish such authentication is for application 60 to determine if the licenses can be trusted. In other words, application 60 has the capability to verify and validate the cryptographic signature of digital signature 52c, or other identifying characteristic of the license. During the activation step described above, both client environment 30 and license server 50 receive a set of keys in a tamper-resistant software “package” that also includes other components, such as the necessary components for activated client environment 30 to verify signature 52 of license 52 in a known manner. Of course, the example above is merely one way to effect a DRM system. For example, the license and content can be distributed from different entities. Also, rights offer 40 can be associated with content by a party other than the party preparing the content. Also, clearinghouse 90 can be used to process payment transactions and verify payment prior to issuing a license.

[0048] For any set of rights, there are two kinds of entities involved, the “supplier” and the “consumer”. The function of the supplier is to offer , and possibly grant, the rights, and the function of the consumer is to select, and possibly exercise the rights. Both the supplier and consumer may actually represent two or more entities. In general, multiple entities may collectively make an offer and grant rights to multiple entities. The supplier and consumer represent any two entities in the content value chain that have a direct relationship with each other regarding the granting of rights. At the beginning of the value chain, the supplier and consumer may be author and publisher. Going down along the value chain, the supplier and consumer may be a publisher and another publisher (for content aggregation), a publisher and

distributor (for content distribution), a distributor and another distributor (for multi-tier content distribution), a distributor and a retailer (for content retailing), a retailer and a consumer (for content consumption), and a consumer and another consumer (for content super-distribution or personal lending).

[0049] An “offer of rights” or “rights offer” expresses how a consumer (e.g. a content distributor or user) can acquire a particular instance of content together with its associated usage rights and/or meta-rights. An offer may or may not contain financial terms. An offer is an expression of mere willingness to commerce negotiation and also an expression of willingness to grant on terms stated. An offer may be expressed in the form of a rights label. A “consideration of rights” is a process as part of the rights granting in which the rights consumer has examined the rights being offered and possibly bargained them and associated terms and conditions. A “choice of rights” is a selection of rights and their associated terms and conditions from a rights offer. It indicates the intent of the consumer to accept these rights and the corresponding terms and conditions. For example, selection can comprise selecting one option 44 from label 40. “Customization of rights” is a process as part of the rights granting in which the rights supplier assembles rights and terms and conditions based on a choice of the rights consumer. The output of this process can be a draft license to be accepted by the rights consumer. A “license of rights” is an expression of rights and possibly conditions accepted and agreed upon by the rights supplier and consumer. It is the output of the rights offering and granting process. A license is a grant to exercise the rights that govern the usage (possibly including further distribution) of content or other items.

[0050] As described above, a rights label, such as rights label 40, may contain a number of options 44 allowing the consumer to make a selection and conduct negotiation (if permitted), while license 52 contains rights the consumer has selected and accepted. Note that the accepted rights may include a right to present offers to others or make selections of offers.

[0051] An example of a distribution chain model is illustrated in Fig. 1. The distribution chain includes a content provider 100, distributor 110, and end user 120. Of course content may be prepared in the manner described above. It is assumed that the content has already been prepared in the model of Fig. 1. Fig. 1 is directed to the transfer of content and shows that, in this example, provider 100 may publish content to distributor 110 or receive content for reuse from distributor 110. Distributor 110 may in turn distribute content to user 120 or receive returned content from user 120. User 100 can use content. To further illustrate the potential complexities of multi-tier distribution chains provider 100 can aggregate content from others, distributor 110, can receive content from other distributors for redistribution, and user 120 can share content with the other users. It is clear that there are plural stages in the content life cycle and plural relationships between the various parties. A precise and consistent specification of rights at the different stages of the life cycle and relationships is important and crucial to persistent protection of content in multi-tier distribution and usage.

[0052] Fig. 2 illustrates the flow of rights in the same model, including rights generating, aggregating, issuing, relinquishing, driving, granting, surrendering, delegating and exercising. The model of Fig. 2 includes the same entities, provider 100, distributor 110, and user 120. It can be seen that, with respect to the flow of rights, each party can grant and accept rights. User 120 can grant and accept rights from other users, a process called “delegation”, in this example.

[0053] The model of Fig. 2 covers many specific content publishing, distribution and use relationships. Other models can be derived from on this model by a different consolidation or segregation of the parties. For example, every provider can be a distributor. This is “direct publishing”, which allows individual authors to distribute/sell their content without any intermediate publisher. Further, every consumer can be a potential distributor. This allows consumers to pass content to each other. This includes supper-distribution,

gifting, and personal lending. In a "Web community" and everyone is able to publish, distribute and consume content. "Content aggregation" allows publishers to compose content from other publishers into composite works. Site license and enterprise use allows sharing content among consumers.

[0054] In general, all the rights relationships shown in Figure 2 can be captured by two generic supplier-consumer models, as shown in Figs. 3(a) and 3(b). Fig 3(a) shows a "push" model and Fig. 3(b) shows a "pull" model. In the push model shown in Fig. 3(a), rights supplier 200 initiates the rights offering and granting process by generating an offer and granting the rights to the rights consumer 210. In the pull model shown in Figure 3(b), rights consumer 210 initiates the process by requesting an offer and accepting the rights from the rights supplier 200.

[0055] An architecture of the preferred embodiment for rights offering and granting is shown in Fig. 4. Architecture 400 can be implemented as a combination of computer hardware and software and includes rights supplier component 402, rights consumer component 438 and communication channel 422 linking these two components. For example, communication channel 422 can be Internet, a direct computer to computer connection, a LAN, a wireless connection or the like. Supplier component 402 is associated with the supplier, i.e. the entity making rights available to a consumer who is the entity going to exercise, i.e., consume the rights. The supplier could be the content owner or provider, or could be a distributor or any "middle-man," such as a retailer or operator of a web site. Consumer component 438 is associated with the consumer who could be the ultimate user (i.e., content consumer) or a "middle-man," such as a retailer, whole-seller, or reseller. Keep in mind that the consumer consumes rights and does not necessarily use (i.e. consume) the content. Both supplier component 402 and consumer component 438 can embody any type of hardware devices, and or software modules, such as a personal computer, a handheld computer, a mobile phone a server, a network, or any combination of the same. Supplier component 402 generates

rights label 40 as offers, presents draft licenses and grants license 52 to the consumer. Consumer component 438 issues requests, select choices of options 44 from rights labels 40, generates counter offers, and accepts licenses 52. Supplier component 402 and consumer component 438 can be embodied in the same device(s) and communication channel 422 can be an internal channel.

[0056] Supplier component 402 contains user interface module 404, communication interface module 420 identity module 406 repository 412 for supplier's rights (e.g., in the form of issued licenses) and database 414 for management related information. User interface 404 accomplishes presentation to the user of the component functions and acceptance of user interactions in a known manner. Communication interface 422 provides the proper formatting and protocols for messages between supplier component 402 and consumer component 438. Identity module 406 ensures that the identity of supplier component 402 can be authenticated by consumer component 438 and may contain authentication information like a password, cryptographic keys or biometric information of the user of supplier component 402. Rights repository 412 stores rights granted to the user of supplier component 402 and may include functions for indexing, searching and updating the rights stored within. Management database 414 is used to archive information generated during the rights offering and granting processes. Such information includes information related to initial offers, consumer choices, possible counter-offers, agreements and final licenses.

[0057] Consumer component 438 includes user interface module 428, communication interface module 424, identity module 426, repository 434 for consumer's rights (e.g., in the form of issued licenses), and database 436 for management related information. User interface 424 deals handles presentation to the user of the component and acceptance of user interactions. Communication interface 422 provides the proper formatting and protocols for rights offering and granting messages between supplier

component 402 and consumer component 438. Identity module 426 ensures that the identity of the consumer component 438 can be authenticated by supplier component 402 and may contain authentication information like a password, cryptographic keys or biometric information of the user. Rights repository 434 stores rights granted to the user of consumer component 438 and may include functions for indexing, searching and updating the rights stored within. Management database 436 is used to archive information generated during the rights offering and granting process. The information includes that related to offers 44, consumer choices, possible counter-offers, agreements and licenses 52. Note that database 436 can store information that is the same as or different from database 414 because the parties may interact with other parties and thus have different archived information.

[0058] Supplier component 402 also includes offer generator module 408 for generating offers, rights composer module 410 for composing licenses, offer templates module 418 for providing templates for generating offers based on previous transactions and common formality of offers, and consumer profiles module 416 for customizing and granting rights based on past consumer characteristics and relationships.

[0059] Consumer component 438 also includes offer analyzer module 430 for understanding rights and their terms and conditions presented within offers, a choice maker module 432 for selecting favorable options specified in offers, a supplier preference module 438 for describing any preferred suppliers based on past and existing supplier characteristics and relationships, and choice patterns module 440 for providing patterns and interests in selection options in offers. For example, the choice pattern module 440 may include a list of preferred suppliers or a list of lowest prices for the item of interest to the consumer. Offer analyzer module 430 and choice maker module 432, respectively, may be combined into one module.

[0060] The process of offering and granting rights within architecture 400 is based on protocols followed by supplier component 402 and consumer

component 438. These protocols generally consist of an offer and acceptance of that offer. Specifically, the protocols include an offering of rights by one party to another and acceptance of that offer by the person to whom it is made. An offer, once made, may be styled so that it may be revoked before acceptance or the offeror could style it so that it cannot be revoked at all or only under certain circumstances definable by the offeror. An offer can also expire in various ways, for example if a deadline for acceptance passes. If there is no specified deadline, then the offer could expire in a predetermined reasonable time, depending on the subject matter of the offer. For periodically available content such as magazines, journals, and even newspapers, a reasonable time could be accorded to the period of the content publication, for example. For dynamically generated or provided content such as streaming content, a reasonable time could be any time before the availability of the content. The rights supplier can dictate other terms of the acceptance, to which the rights consumer is bound. For example, the offer may require acceptance in sending back in a certain form via an email or through a certain web page interface.

[0061] Fig. 5(a) illustrates the workflow of protocol 500 of a push model for rights granting. Supplier component 402 generates an offer of rights in the form of rights label 40 for example, with possibly many options 44, and sends it to consumer component 438 (510). Consumer component 438 considers the offer and its possible options, and responds to supplier component 402 with a choice of any of the optional rights offer 44 (512). Supplier component 402 customizes rights according to the consumer's response, and issues the rights to the user of consumer component 432 (514) in the form of a draft license.

[0062] Consumer component 438 then accepts the draft license if it corresponds to the choice made and is otherwise acceptable (516). Upon acceptance, supplier component 402 generates license 52 and transmits license 52 to consumer component (518). Keep in mind that grant 52b of

license 52 can include usage rights and/or meta-rights. Therefore license 52 can permit the user of consumer component 438 to grant rights to others in a similar fashion. However, the derivable rights are controlled by upstream parties through the use of meta-rights. Additionally, the protocol can include steps where supplier component 402 requests to make payment through a credit card of the user of consumer component 438, and the user component 402 provides the information and authorizes the charge. Both supplier component 402 and consumer component 438 can generate status reports on success or failure of the process. Further, parties can authenticate each other during the process and maintain authentication through the process.

[0063] Fig. 5(b) shows a protocol of pull model for rights granting. First, consumer component 438 sends a request to supplier component 402 to indicate an interest in obtaining certain rights in content (520). Supplier component 402 then responds with an offer, in the form of label 40 having plural offer options 44, covering the rights requested by consumer component 438, and sends the offer to consumer component 438 (522).

[0064] Consumer component 438 then considers the offer and its options, and responds to supplier component 402 with a choice of one of the offer options (524). Supplier component 402 customizes rights according to the response, and grant the rights to the consumer in the form of a draft license (526). Consumer component 438 then accepts the draft license (528) and supplier component 402 issues license 52 granting rights to consumer component 438 (530). Once again the rights can include meta-rights.

[0065] Fig. 6 illustrates the offer generation process 600 performed by offer generator module 408 in supplier component 402. In offer generation process 600, available rights are first collected in block 602. Rights may be available from a previous supplier by being derived from meta-rights granted to the supplier or may be originally created rights. In step 604 it is determined whether supplier has a right to make an offer to the consumer. For example, if the consumer is known to be a minor and the content is restricted to an

adult consumer or if the consumer is on a list of those prohibited from receiving content, the supplier may not make an offer. In such case, the offer generation process terminates in step 606. If the supplier has the right to make an offer, the process then determines all the rights that can be offered to the consumer in step 608 by parsing the rights collected in step 602. Next, in step 610, the process determines whether the consumer has requested any specific rights. If a request has been received, the process further filters the determined rights that can be offered, taking the received consumer requested rights into consideration and comparing them to the available rights. Then, the process determines whether an offer template needs to be applied in steps 614.

[0066] For example, the consumer might be offered standard rights included in the template, such as printing right, archiving right, etc. of the content. If an offer template is available and needed, the offer template is then applied in steps 616. In steps 618, human intervention may be provided to further make adjustments to the offer template or to any of the rights that are available for offering thus far in the process. Next, restrictions can be applied, through conditions and/or state variables. For example, a time restriction may be placed on certain rights in step 620. Finally, a digital signature or other authentication is provided with the collection of rights to be offered in step 622 and an authenticated offer, in the form of rights label 40 is made in step 624 and presented to consumer component 438 in step 624.

[0067] Fig. 8 illustrates rights customization process 800 which is performed by rights composer module 410 in supplier component 402. Initially, consumer's choices are received in step 802. Choices are rights and conditions of an option 44 selected label 40 of step 624 (Fig. 6). The process then determines if supplier component 402 has the right to grant rights to consumer component 438 in step 804. For example, if the consumer fails to meet a certain requirement, such as minimum age or proof of residence in a locale where content may be licensed, for example, granting a license may

not be proper, and the rights customization process 800 terminates in step 806. Otherwise, consumer selected choices are analyzed in step 808 to ascertain if they are discernible by supplier component 402. For example, the choices can be parsed to see if they are understandable.

[0068] Next, the process determines if consumer information is available in step 810. For example, consumer profiles may be stored in database 414 (Fig. 4). If available, the consumer information is taken into consideration in step 812 for further analysis of consumer choices. In step 812, dynamic information can also be considered as described below. For example, the profile may include a trust rating or address of the consumer that renders it desirable or undesirable to provide certain rights. The process then determines if the choices are reasonable in step 814. This determination may be carried out, for example, computationally or with human intervention. If the customer's choices are deemed unreasonable, re-negotiation of the customer's choices is then performed in block 816. In this re-negotiation process, the customer is presented with a new proposed offer based on the previously analyzed choices, the customer is given an opportunity to submit new choices offered, and the right customization process 800 begins again in step 802. Otherwise, a license including the selected rights is created in step 818.

[0069] After a license is created, if consumer acceptance is necessary (step 820), it is presented to the consumer for review in step 822. If the consumer does not agree with the terms in the license in step 824, re-negotiation is then initiated in step 816, which re-starts the rights customization process 800 again in step 802. In step 820, if a review by the consumer is not required, then the license is authenticated in step 826 to create a completed license 52 in step 828 which is to be issued and associated with content 42.

[0070] Fig. 7 illustrates offer consideration process 700 which is performed by offer analyzer module 430 and choice maker module 432 of consumer component 438. Available offers are first collected in step 702. In step 704,

process 700 determines whether it has a right to accept offers from the supplier. For example, if the consumer certain restrictions on the purchase of content, such as an age restriction or a restriction against accepting content from outside an enterprise, the consumer may not accept an offer. In such a case, the offer consideration process terminates in step 706. If the consumer has the right to accept offers from the supplier, the offers are then analyzed in step 708 to ascertain if they are discernible. If it is determined that supplier preferences are available in step 710, the offers are filtered in step 712 based on the preferences. For example, the consumer may trust a specific supplier, or otherwise prefer transactions with that supplier, more that other suppliers. Next, step 714 determines if consumer preferences are available and, if so, they are applied in step 716 to the offers. Once all the offers are analyzed, by applying the logic of steps 708-714 and any other desired logic, the consumer then selects options in block 718 and specifies contingencies in block 720. The selection of options can be done automatically. If human intervention is desired, the customer can intervene and further specify additional choices or conditions desired. Any preferences, rules, or other logic can be used to analyze offers.

[0071] Overall, as can be seen in the description of Figures 6, 7, and 8 above, the consumer sends a request, and then a license is constructed. Either the supplier or the consumer could draft the content of the license, but in the example above the supplier does so. The request is a subset of an offer and the offer has one or more options. The supplier makes the offer available to the consumer sending the request (and to other consumers if that is the desire), and the consumer (including other consumers, if applicable) makes choices. Then, the supplier analyzes the choices, and constructs the license (i.e. a grant of rights). Note that the request can also be rejected, or a counter proposal could be made and the same process could then repeat for the counter proposal.

[0072] Also, when the supplier analyzes the request, the analysis may be done automatically, or with human intervention. When the consumer considers the offer, the choice or acceptance may be done automatically, or with human intervention. Either the offer or a license, or both, may be generated based on the dynamic information, the consumer's information, and the consumer's request, such as described above.

[0073] The dynamic information may include many kinds of information including information related to pricing, status of the network, the traffic of a web site at each moment of time, discounts given, coupons given, the habits of the consumer, how many times the content has been used, for how long the content was used, where it was used, or the like. The dynamic information can be tracked as state variables and the values of the state variables can be checked and updated as necessary.

[0074] Dynamic information is information capable of being (although, it need not actually be) changed or created by or by reference to a non-static element. For example, the dynamic information can be obtained based on a formula, database, curve, predetermined table, percentage of a value, a function, reference to other data, such as the prime rate of interest or the change in a stock market index, and/or by a human intervention of the user or distributor, and/or consumer's input.

[0075] The consumer's information may include information such as the age of the consumer, the credit history of the consumer, the credit limit of the consumer, income of the consumer, what kind of rights or licenses obtained, the password of the consumer, the key assigned to the consumer, club membership for access or discount, the class of the consumer based on a predetermined criteria, or any other data, identification characteristics and information. The supplier's information may include some or all of the subjects of information as the consumer's information, and may also include, for example, available options or variations, suppliers, shipping information, and other information.

[0076] The system and processes disclosed in this invention support multi-tier and super distributions of content. The following is a use case that shows how this can be modeled and supported. It illustrates the process of offering and granting rights by showing the process of transforming offered rights to a rights supplier (the content distributor in this case) to granted rights to a rights consumer (the end user in this case). It specifically shows how an offer is generated from an existing license, how this offer is considered with a choice, and how a final license is issued. Meta-rights provide a mechanism for permitting the transfer of rights from one party to the next party in a content distribution chain.

[0077] Suppose that a content provider P of some content C wants to specify that a distributor D may sell, to any end user within the region of the United States (US), the “play” right at a flat rate of \$1 and the “print” right at a cost of \$4 per copy (both are paid by D to P). The provider also allows the content distributor to add its own conditions to the “play” and “print” rights it issues to end users.

[0078] A license from the content provider to the distributor may resemble the following using the XrML rights language.

```

<license>
  <grant>
    <forAll varName="user"/>
    <forAll varName="distributorConditionForPlay"/>
    <principal id="distributor"/>
    <issue/>
    <grant>
      <principal varRef="user"/>
      <play/>
      <digitalResource licensePartId="book"/>
      <allCondition>
        <region regionCode="US"/>
        <condition varRef="distributorConditionForPlay"/>
      </allCondition>
    </grant>
    <fee>
      <flat currencycode="USD">1</flat>
      <to licensePartId="provider"/>
    </fee>
  </grant>
  <grant>
    <forAll varName="user"/>
    <forAll varName="distributorConditionForPrint"/>
    <principal id="distributor"/>
    <issue/>
    <grant>
      <principal varRef="user"/>
      <play/>
      <digitalResource licensePartId="book"/>
      <allCondition>
        <region regionCode="US"/>
        <condition varRef="distributorConditionForPrint"/>
      </allCondition>
    </grant>
    <fee>
      <perUse regionCode="USD">5</perUse>
      <to licensePartId="provider"/>
    </fee>
  </grant>
  <issuer id="provider"/>
</license>

```

[0079] The distributor may make an offer to the end user based on the rights it has as expressed in the license above. Note that usage rights and conditions of each option are set forth as XML elements between <grant> tags. In the following offer, note that the distributor adds a fee condition for getting the “play” right, charging the end user \$2 (\$1 more than it pays to the

provider), and another fee condition for the “print” right, charging the end user \$6 per print copy (\$1 more than it pays to the provider). The distributor also limits the offer to an acceptance time period (up to December 31, 2002). Meta rights granted to the distributor permit the distributor to modify the grant in the license, as described above, and make the offer.

```

<offer>
  <grant>
    <forAll varName="user"/>
    <principal varRef="user"/>
    <obtain/>
    <grant>
      <principal varRef="user"/>
      <play/>
      <digitalResource licensePartId="book"/>
      <region regionCode="US"/>
    </grant>
    <fee>
      <flat currencyCode="USD">2</flat>
      <to licensePartId="distributor"/>
    </fee>
  </grant>
  <grant>
    <forAll varName="user"/>
    <principal varRef="user"/>
    <obtain/>
    <grant>
      <principal varRef="user"/>
      <print/>
      <digitalResource licensePartId="book"/>
      <allCondition>
        <region regionCode="US"/>
        <fee>
          <perUse currencyCode="USD">6</perUse>
          <to licensePartId="distributor"/>
        </fee>
      </allCondition>
    </grant>
  </grant>
  <issuer id="distributor">
    <validityInterval>
      <until>2002:12:31</until>
    </validityInterval>
  </issuer>
</offer>

```

[0080] When the offer is presented to an end user, the end user may choose to get only the right to “play” for the flat fee of \$2 and responds to the

distributor with a choice set forth as an XML element between <choice> tags as follows.

```

<choice>
  <grant>
    <principal id="anEndUser"/>
    <obtain/>
    <grant>
      <principal id="anEndUser"/>
      <play/>
      <digitalResource licensePartId="book"/>
      <region regionCode="US"/>
    </grant>
    <fee>
      <flat currencyCode="USD">2</flat>
      <to licensePartId="distributor"/>
    </fee>
  </grant>
  <issuer id="anEndUser">
    <validityInterval>
      <until>2002:12:31</until>
    </validityInterval>
  </issuer>
</choice>

```

[0081] Note that the request can also be rejected. Note also that a response can also be constructed as a counter offer for rights not originally offered by the distributor. When the distributor receives the choice from the end user, it then issues a license to the user as shown below.

```

<license>
  <grant>
    <principal id="anEndUser"/>
    <obtain/>
    <grant>
      <principal id="anEndUser"/>
      <play/>
      <digitalResource licensePartId="book"/>
      <region regionCode="US"/>
    </grant>
    <fee>
      <flat currencyCode="USD">2</flat>
      <to licensePartId="distributor"/>
    </fee>
  </grant>
  <issuer id="distributor">
    <issuedTime>
      2002:05:06
    </issuedTime>
  </issuer>
</license>

```


[0082] Note that in all the XML documents above, the issuers may choose to digitally sign the documents using some digital signature algorithms. The recipients of these documents have options to verify the validity of these documents by checking the validity of the attached digital signatures. Access to the various documents, and elements thereof, can be controlled using known techniques.

[0083] In some situations offering and granting result in a license with a fresh state for content usage. As one starts to exercise the rights, derived rights, obtained as a result of meta-rights, may inherit and/or share the state variable values associated with the rights. For example, when one is granted with the right to print 5 times and make 4 copies of some document, all new copies may have the same set of rights but share the state (or remaining rights) with the original. After the original has been printed 2 times and a new copy was then made, the copy and original can all together print 3 times and make 2 more new copies.

[0084] Thus, the exemplary embodiments include a method for transferring usage rights adapted to be associated with items. The method includes generating, by a supplier, at least one first offer containing usage rights and meta-rights for the item, the usage rights defining a manner of use for the items, the meta-rights specifying rights to derive usage rights or other meta-rights, presenting the offer to a first consumer, receiving a selection from the first consumer indicating desired usage rights and meta-rights, and generating a first license granting the desired usage rights and meta-rights to the first consumer. The exemplary embodiments further include a system for transferring usage rights adapted to be associated with an item to be licensed in multi-tier channels of distribution with downstream rights and conditions assigned at least one level. The system includes a supplier component, comprising a supplier user interface module, an offer generator module for generating an offer containing at least usage rights and of meta-rights, a rights composer module for composing a draft license, and a repository for

supplier's rights, a supplier management database. The system further includes a consumer component comprising a consumer user interface module, an offer-consideration module configured to analyze the offers generated by the supplier component and select offers based on the analysis, and a repository for consumer's rights, a consumer management database. The exemplary embodiments still further include a method for generating a license to digital content to be used within a system for at least one of managing use and distribution of the digital content. The method includes presenting a consumer with an offer including meta-rights, receiving a selection by the consumer of at least one meta-right in the offer, generating a license based on the selection, wherein the license permits the consumer to exercise the at least one meta-right and permits the consumer to offer at least one derived right derived from the at least one meta-right and generate a license including the at least one derived right.

[0085] Fig. 12 illustrates an exemplary system including a common state-of-rights server, according to the present invention. In FIG. 12, the exemplary system can include a common state-of-rights server of the system 1201, including a state-of-rights manager 1209, and one or more state-of-rights repositories 1214, and one or more license servers 1200, including a meta-rights manager 1210, a usage rights manager 1212, an authorization component 1208, a condition validator 1206, a state-of-rights manager 1204, one or more state-of-rights repositories 1216, a license manager 1203, a license interpreter 1202, and one or more license repositories 1218.

[0086] The common state-of-rights server 1201 can be configured as a remote server connected with one or more of the license servers 1200. The common state-of-rights server 1201 provides comparable services as the state-of-rights manager 1204 in the license servers 1200 via the state-of-rights manager 1209. The services provided by the state-of-rights server 1201 are accessible and states that the server 1201 manages can be shared by one or more rights suppliers and rights consumers (not shown).

[0087] The state-of-rights server 1201 can be configured as a remote server connected with one or more of the license servers 1200 via one or more communication links 1220, and the like. The services provided by the state-of-rights server 1201 also can be integrated within one or more of the license server 1200 and such services can be accessible by other rights suppliers, rights consumers, and the like.

[0088] The license manager 1203 derives new rights based on an offer, which can include any suitable machine-readable expression, and optionally including meta-rights. While deriving rights, the license manager 1203 can create new state variables to be associated with derived rights. The creation of state variables and their scopes can be prescribed in the offer or by some other function in the system. The state variables can be created in one or more instances, for example, prior to rights derivation, during rights derivation, upon fulfillment of conditions, during a first exercise of rights associated with the state variables, and the like. The state variables can be designated exclusively for a specific rights consumer, can be shared among rights consumers, and can be shared among rights consumers and other entities, such as rights suppliers, and the like. The license manager 1203 can interact with the state-of-rights manager 1204 to associate new state variables with physical addresses in one or more of the state-of-rights repositories 1216. The state-of-rights manager 1204 can access the one or more state-of-rights repositories 1216 and can interact with the state-of-rights server 1201 to access shared state variables from one or more of the state-of-rights repositories 1214.

[0089] Designated state variables can be used to support a license that grants a recipient of the license a right to print content 5 times, shared state variables can be used to support a site license that grants a group of authorized users a right to print content an aggregated total of 100 times, and the like. A designated state variable can be updated when the corresponding right is exercised, whereas a shared state variable can be updated when an

authorized user exercises the corresponding right. In other words, a shared state variable can include a data variable that is updated in response to actions by a plurality of users and which is globally applied to each of the users.

[0090] There are multiple ways to specify the scope of state variables, each of which can affect whether the derivative state variables can be shared, how the derivative state variables can be shared, and the like. For example, a state variable can be local, and solely confined to a recipient or can be global, and shared by a predetermined group of recipients. A global state variable can be shared by a group of recipients not determined when derived rights are issued, but to be specified later, perhaps based on certain rules defined in the license or based on other means. A global state variable can be shared between one or more rights suppliers, predetermined recipients, un-specified recipients, and the like. Advantageously, depending on the sharing employed with a given a business model and the rights granted in the meta-rights, state variables can be created at different stages of the value chain.

[0091] A set of non-exhaustive exemplary usages of state variables will now be described. For example, a state variable can be unspecified in meta-rights, which means the identifier and value of the state variable are yet to be determined by the meta-rights manager module 1210 and included in the derived right. If a distinct state variable is assigned to each derived right, the scope of the state variable in the derived right is typically exclusive to the recipient.

[0092] Fig. 13 is used to illustrate employing of a state variable in deriving exclusive usage rights, according to the present invention. In Fig. 13, rights 1302 and 1303 derived from an offer 1301 are exclusive to each respective consumer. The offer 1301 is a type of meta-right of which the recipients have the rights to obtain specific derivative rights when the conditions for obtaining such rights are satisfied. Accordingly, the exemplary offer 1301 has an unspecified state variable 1304. However, specific state variable 1305 and

1306, each with uniquely assigned identifications (IDs) are included in the derived rights 1302 and 1303. The derived state variables 1305 and 1306 are bound to their associated derived rights, e.g., "AlicePlayEbook" (i.e., Alice has the right to play Ebook) is bound to derived right 1302, and "BobPlayEbook" (i.e., Bob has the right to play Ebook) is bound to derived right 1303. The "AlicePlayEbook" variable can be updated when Alice exercises her play right, whereas the "BobPlayEbook" variable can be updated when Bob exercises his play right.

[0093] Other than deriving rights from an offer, a right can transfer from an entity to a recipient. When a right is transferred, the governing of the associated state variable is also transferred to the recipient. After a right is transferred, the source principal typically can no longer exercise the right, whereas the recipient can exercise the right. The license server governing the exercising of a right of a recipient assumes the responsibility for state management. If, however, the state variables are managed by the common state of right server 1201, the state of right server 1201 needs to be informed of the transfer of right. Specifically, the state variable can be managed in the context of the recipient after the transfer of right.

[0094] When a right is to be shared between the source principal and the recipient, the associated state variable is referenced in the derived right. If the same right is shared with multiple recipients, then typically all of the recipients share the same state variables with the source principal. In this case, a shared state can be managed by an entity that is accessible by all sharing principals.

[0095] Fig. 14 is used to illustrate employing of a state variable in deriving inherited usage rights, according to the present invention. In Fig. 14, a derived right can inherit a state variable from meta-rights. For example, a personal computer (PC) of a user, Alice, can be configured to play an e-book according to a license 1403. A personal data assistant (PDA) of Alice also can obtain a right to play the e-book according to offer 1401, if the PC and

PDA share the same state variables 1404 and 1405, e.g., "AlicePlayEbook." A derived right 1402 allows Alice also to play the e-book on her PDA as long as the PDA and the PC share a same count limit 1406 of 5 times.

[0096] When a usage right is to be shared among a predetermined set of recipients, a state variable for tracking a corresponding usage right can be specified in a meta-right using a same state variable identification for all recipients. During a process of exercising the meta-right, the same state variable identification is included in every derived right.

[0097] Fig. 15 illustrates the use of state variable in deriving rights that are shared among a known set of rights recipients, according to the present invention. In Fig. 15, a site license 1501 is issued to FooU university. For example, via the site license 1501, a librarian is granted a right to issue rights that allow FooU students to play, view, and the like, corresponding content, such as e-books and the like, as long as such usage is tracked by a state variable 1504, e.g., "www.fooU.edu." Accordingly, rights 1502 and 1503 derived from the site license 1501 include state variables 1505 and 1506, "www.fooU.edu," which can be updated when corresponding students, Alice and Bob, play the e-book.

[0098] When a usage right is to be shared among a dynamic set of recipients, the state variable can stay unspecified in the usage right. When exercising a meta-right and a set of recipients is known, a state variable can be specified using some identification unique to the known recipients and can be included within a derived right.

[0099] Fig. 16 is used to illustrate employing of a state variable in deriving rights that are shared among a dynamic set of rights recipients, according to the present invention. In Fig. 16, an offer 1601 specifies that a distributor can issue site licenses to affiliated clubs, allowing 5 members of each club to concurrently view, play, and the like, content, such as an e-book. A corresponding state variable 1607 associated with such a right can be unspecified in the offer 1601. When corresponding rights 1602 and 1603 are

issued to affiliated clubs, the corresponding club identities are used to specify state variables 1608 and 1609 in the issued rights. The offers 1602 and 1603 are meta-rights derived from the offer 1601, with offer being assigned the distinct state variables 1608 and 1609. Further rights 1604-1606 can be derived from the offers 1602 and 1603 to be shared among members of each respective club. The licenses 1604 and 1605 are examples of rights derived from the offer 1602, and which inherit the state variable 1608, e.g., "urn:acme:club," whereas the license 1606 inherits the state variable 1609, e.g., "urn:foo:club."

[00100] Not only can state variables be shared among principals, such as rights suppliers, consumers, and the like, a state variable can be shared among multiple exercisable rights. Fig. 17 is used to illustrate employing of a state variable for maintaining a state shared by multiple rights, according to the present invention. In Fig. 17, a same state variable 1703 is associated to both a right to print 1702 and the right to play 1701, so that the total number of playing, printing, and the like, can be tracked together.

[00101] The state of rights can depend on more than one state variable. Fig. 18 is used to illustrate employing of multiple state variables to represent one state of rights, according to the present invention. The example described with respect to Fig. 18 builds upon the example described with respect to Fig. 16. In FIG. 18, a usage right can be tracked by employing multiple state variables 1807 and 1808 in an offer 1801. The state variable 1808, for example, representing a priority level, can stay unspecified in the corresponding offers 1802 and 1803 (e.g., site licenses). The corresponding state variables 1809-1811, for example, used for setting a priority, can be assigned to each member in the corresponding licenses 1804, 1805 and 1806. The corresponding right to view, play, and the like, can now be dependent on two state variables, effectively restricting 5 simultaneous views, plays, and the like, per priority level.

[00102] One state variable can represent a collection of states. For example, a unique identification can be used to represent a state variable, and an appropriate mechanism can be employed to map such unique id to a database of multiple variables, where each variable represents a distinct state.

[00103] The scope of state variables can be used to determine entities by which the state variables can be managed. For example, for a local state variable, usage tracking of associated rights thereof can be managed solely by a trusted agent embedded within a rights consumption environment, such as a media player, and the like. In addition, such usage tracking can be conducted by a trusted remote service, such as the common state-of-rights server 1201. Further, shared global state variables can be made accessible by multiple trusted agents. To avoid privacy issues, security issues, trust issues, rights issues, and the like, associated with accessing content, such as data, and the like, included within a peer rights consumption environment, managing of such shared global state variables can be performed by a remote service, such as the state-of-rights server 1201.

[00104] A counter is a common form of state variable usage. For example, such state sharing can include counter sharing where a state represents a number of times a right has been exercised, an event has occurred, and the like. Such counter sharing can be manifested in various forms and occur in many contexts, such as: tracking a number of simultaneous uses, tracking a number of sequential uses, sequencing (e.g., a commercial must be viewed before free content can be accessed), a one-time use constraint, a transaction count, a delegation control level, a super-distribution level, dependency on at least one or more services or devices, and the like.

[00105] In addition, state variables can be incarnated in a wide variety of forms. For example, a state variable can be used to track specific time slots within a period of time, such as used by a movie studio to transfer syndication

rights to a specific TV station, to transfer syndication rights shared by a group of stations, to transfer syndication rights assigned through a bidding process, and the like.

[00106] State variables also can be employed, for example, with regional selling or distribution rights, in a statement from a financial clearing house to acknowledge that an appropriate fee has been paid, as a status of whether a commercial has been watched before free content can be accessed, and the like.

[00107] Not all rights need be associated with states. Fig. 19 is used to illustrate a case where not all rights are associated with states, according to the present invention. In Fig. 19, an offer 1901 allows a user, Alice, to grant an unlimited play right, view right, and the like, to her PDA. Such a play right need not be associated with any state. Accordingly, derived right 1902 also has an unlimited play right to the content, as well as the right 1903 for her PC.

[00108] Not all rights which are associated with states are shared or inherited. For example, some rights are meant for off-line usage, can be transferred in whole to another device, and hence are not shared with other devices. Fig. 20 is used to illustrate a case where not all rights which are associated with states are shared or inherited, according to the present invention. In Fig. 20, even though a play right 2003 of a user, Alice, a play right 2002 of a PDA of Alice, and a play right 2003 of a PC of Alice specify a same state variable identification 2004, a same state need not be shared since each device can track a state thereof locally. Advantageously, such an implementation would allow the PC and the PDA to each play the corresponding content up to 5 times.

[00109] Fig. 21 illustrates a form of an offer which does not explicitly include meta-rights. In Fig. 21, an offer 2101 is configured as a site license written in English. Licenses 2102 and 2103 are instances derived from the offer 2101. In an exemplary embodiment, variables 2104 and 2105 can be

created based on interpretation of the offer 2101, for example, by the system of Fig. 12.

[00110] The preferred embodiment can utilize various devices, such as a personal computers, servers, workstations, PDA's, thin clients, and the like. For example, the client environment can be a handheld device such as a mobile phone or a PDA. Various channels for communication can be used. Further, the various functions can be integrated in one device. For example, the license server function can be accomplished by software within the client environment. Further, the function of the license server or other modules for making offers, selecting rights and granting licenses can be accomplished in the same device. The disclosed functional modules are segregated by function for clarity. However, the various functions can be combined or segregated as hardware and/or software modules in any manner. The various functions can be useful separately or in combination.

[00111] The various elements and portions thereof can be stored on the same device or on different devices. For example, a license can be stored together with, or separate from, content. Further, the various elements of a license can be stored on separate devices. For example the values of state variables can be stored in a state variable repository of a system that tracks the current value of state variables. Various links, references, specifications, and the like can be used to associate the elements.

[00112] The invention has been described through exemplary embodiments and examples. However, various modifications can be made without departing from the scope of the invention as defined by the appended claims and legal equivalents.

What is claimed is:

1. A method for sharing rights adapted to be associated with items, the method comprising:

generating at least one of usage rights and meta-rights for the items;

defining, via the usage rights, a manner of use for the items; and

defining, via the meta-rights, a manner of rights transfer for the items;

wherein the usage rights or the meta-rights include at least one state variable that is shared by one or more rights.

2. The method of claim 1, wherein the state variable inherits a state thereof for content usage or rights transfer from other generated usage rights and meta-rights.

3. The method of claim 1, wherein the state variable shares a state thereof for content usage or rights transfer with other generated usage rights and meta-rights.

4. The method of claim 1, wherein the state variable inherits a remaining state for content usage or rights transfer from other generated usage rights and meta-rights.

5. The method of claim 1, wherein the state variable is updated upon exercise of a right associated with the state variable.

6. The method of claim 1, wherein the state variable represents a collection of states.

7. The method of claim 1, further comprising deriving at least one right from the generated usage rights and meta-rights, wherein the derived right includes at least one state variable that is shared with or inherited from the

generated usage rights and meta-rights and is used for determining a state of the derived right.

8. The method of claim 7, further comprising a plurality of state variables that determine the state of the derived right.

9. The method of claim 7, wherein the state variable is unspecified in the derived right, is created during a rights transfer, and is assigned to the derived right.

10. The method of claim 7, wherein the state variable is transferred from the generated usage rights and meta-rights to the derived right.

11. The method of claim 7, further comprising generating a license including the derived right.

12. The method of claim 7, further comprising deriving a plurality of rights from the generated usage rights and meta-rights, wherein the state variable is shared by the derived rights.

13. A system for sharing rights adapted to be associated with items, the system comprising:

means for generating at least one of usage rights and meta-rights for the items;

means for defining, via the usage rights, a manner of use for the items;
and

means for defining, via the meta-rights, a manner of rights transfer for the items;

wherein the usage rights or the meta-rights include at least one state variable that is shared by one or more rights.

14. The system of claim 13, wherein the state variable inherits a state thereof for content usage or rights transfer from other generated usage rights and meta-rights.

15. The system of claim 13, wherein the state variable shares a state thereof for content usage or rights transfer with other generated usage rights and meta-rights.

16. The system of claim 13, wherein the state variable inherits a remaining state for content usage or rights transfer from other generated usage rights and meta-rights.

17. The system of claim 13, wherein the state variable is updated upon exercise of a right associated with the state variable.

18. The system of claim 13, wherein the state variable represents a collection of states.

19. The system of claim 13, further comprising means for deriving at least one right from the generated usage rights and meta-rights, wherein the derived right includes at least one state variable that is shared with or inherited from the generated usage rights and meta-rights and is used for determining a state of the derived right.

20. The system of claim 19, including a plurality of state variables that determine the state of the derived right.

21. The system of claim 19, wherein the state variable is unspecified in the derived right, is created during a rights transfer, and is assigned to the derived right.

22. The system of claim 19, wherein the state variable is transferred from the generated usage rights and meta-rights to the derived right.

23. The system of claim 19, further comprising means for generating a license including the derived right.

24. The system of claim 19, further comprising means for deriving a plurality of rights from the generated usage rights and meta-rights, wherein the state variable is shared by the derived rights.

25. The system of claim 13, wherein the means for generating; the means for defining via the usage rights, and the means for defining via the meta-rights comprise at least one of computer-executable instructions, and devices of a computer system.

26. A device for sharing rights adapted to be associated with items, the device comprising:

means for receiving at least one of usage rights and meta-rights for the items;

means for interpreting, via the usage rights, a manner of use for the items; and

means for interpreting, via the meta-rights, a manner of rights transfer for the items;

wherein the usage rights or the meta-rights include at least one state variable that is shared by one or more rights.

27. The device of claim 26, wherein the state variable inherits a state thereof for content usage or rights transfer from other generated usage rights and meta-rights.

28. The device of claim 26, wherein the state variable shares a state thereof for content usage or rights transfer with other generated usage rights and meta-rights.

29. The device of claim 26, wherein the state variable inherits a remaining state for content usage or rights transfer from other generated usage rights and meta-rights.

30. The device of claim 26, wherein the state variable is updated upon exercise of a right associated with the state variable.

31. The device of claim 26, wherein the state variable represents a collection of states.

32. The device of claim 26, further comprising means for deriving at least one right from the generated usage rights and meta-rights, wherein the derived right includes at least one state variable that is shared with or inherited from the generated usage rights and meta-rights and is used for determining a state of the derived right.

33. The device of claim 32, including a plurality of state variables that determine the state of the derived right.

34. The device of claim 32, wherein the state variable is unspecified in the derived right, is created during a rights transfer, and is assigned to the derived right.

35. The device of claim 32, wherein the state variable is transferred from the generated usage rights and meta-rights to the derived right.

36. The device of claim 32, further comprising means for generating a license including the derived right.

37. The device of claim 32, further comprising means for deriving a plurality of rights from the generated usage rights and meta-rights, wherein the state variable is shared by the derived rights.

38. The device of claim 26, wherein the means for receiving, the means for interpreting via the usage rights, and the means for interpreting via the meta-rights comprise at least one of computer-executable instructions, and devices of a computer system.

39. The device of claim 26, wherein one or more of the means for receiving, the means for interpreting via the usage rights, and the means for interpreting via the meta-rights are specified in a license.

ABSTRACT OF THE DISCLOSURE

[00113] A method, system and device for sharing rights adapted to be associated with items, the method and system including generating at least one of usage rights and meta-rights for the items; defining, via the usage rights, a manner of use for the items; and defining, via the meta-rights, a manner of rights transfer for the items. The device including receiving at least one of usage rights and meta-rights for the items; interpreting, via the usage rights, a manner of use for the items; and interpreting, via the meta-rights, a manner of rights transfer for the items. The usage rights or the meta-rights include at least one state variable that is shared by one or more rights.

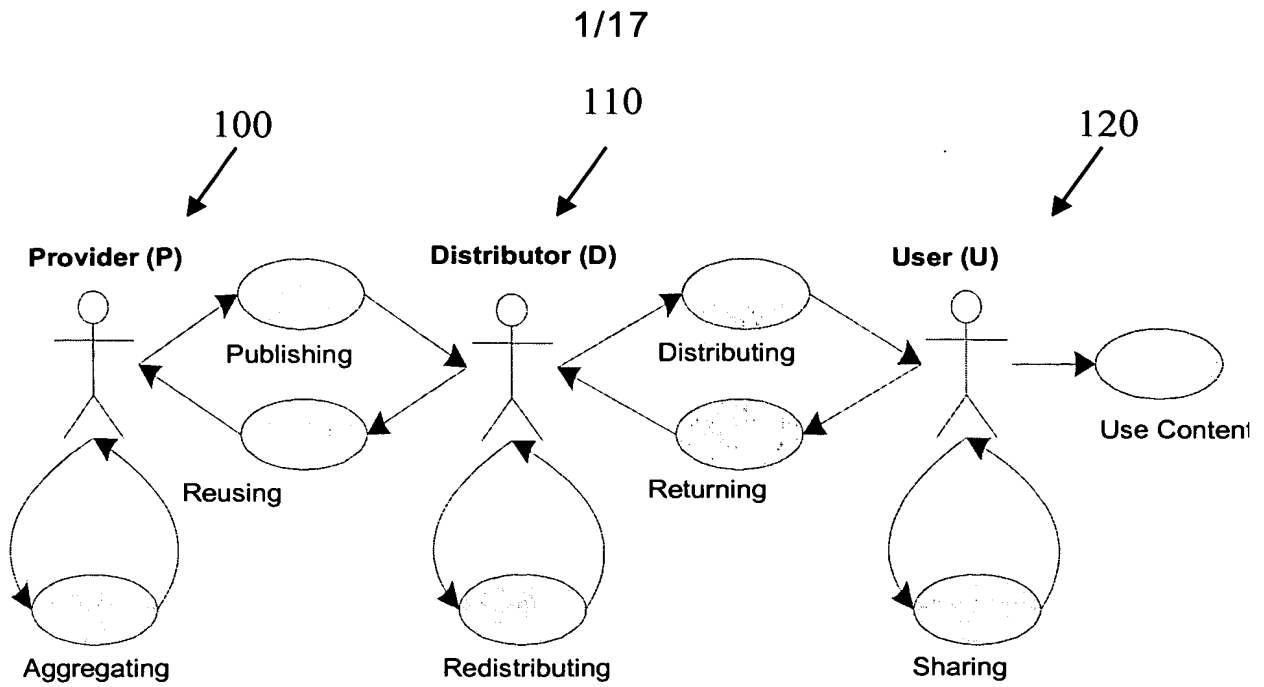


Fig. 1

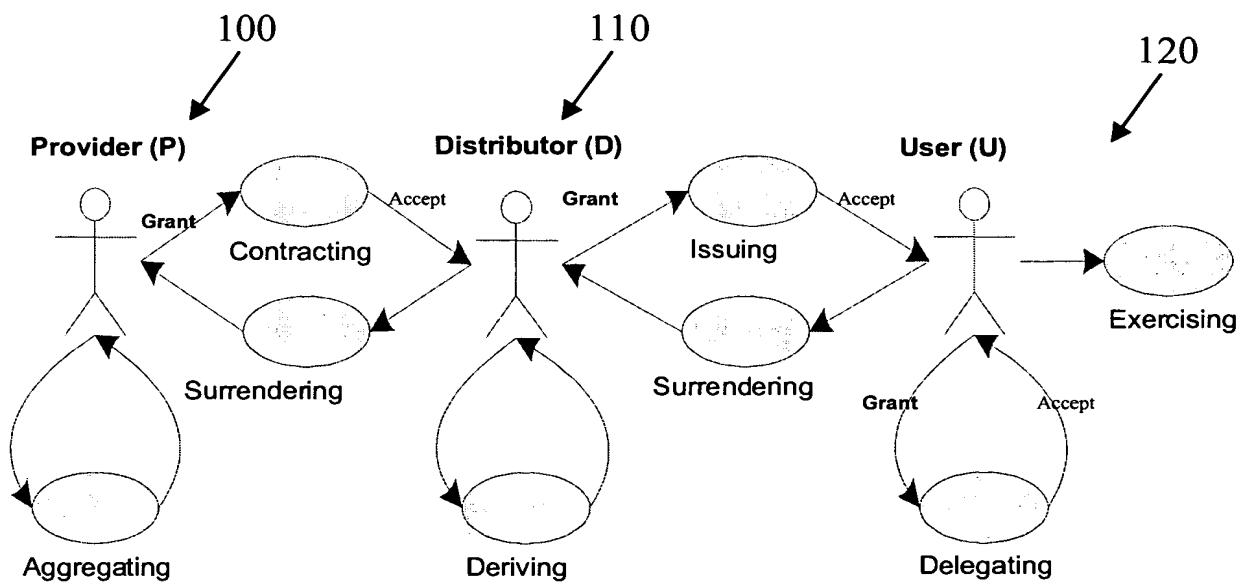


Fig. 2

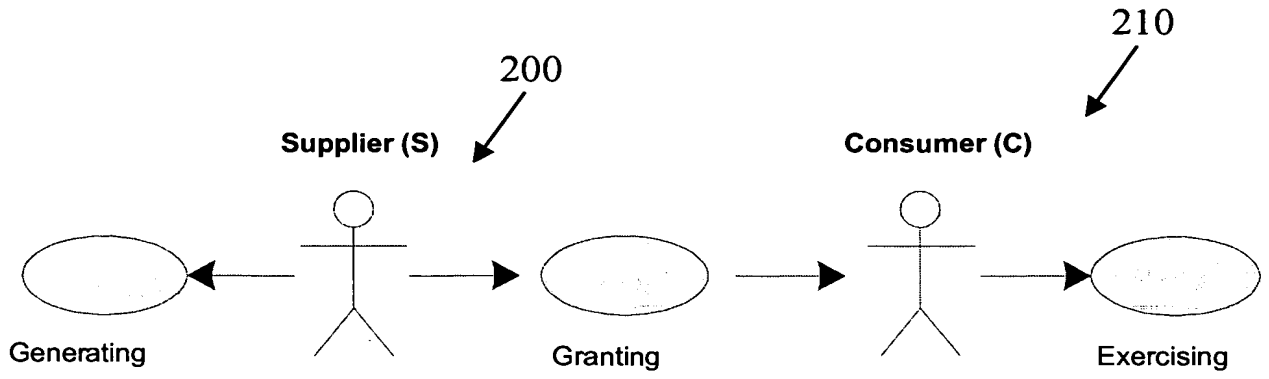


Fig. 3(a)

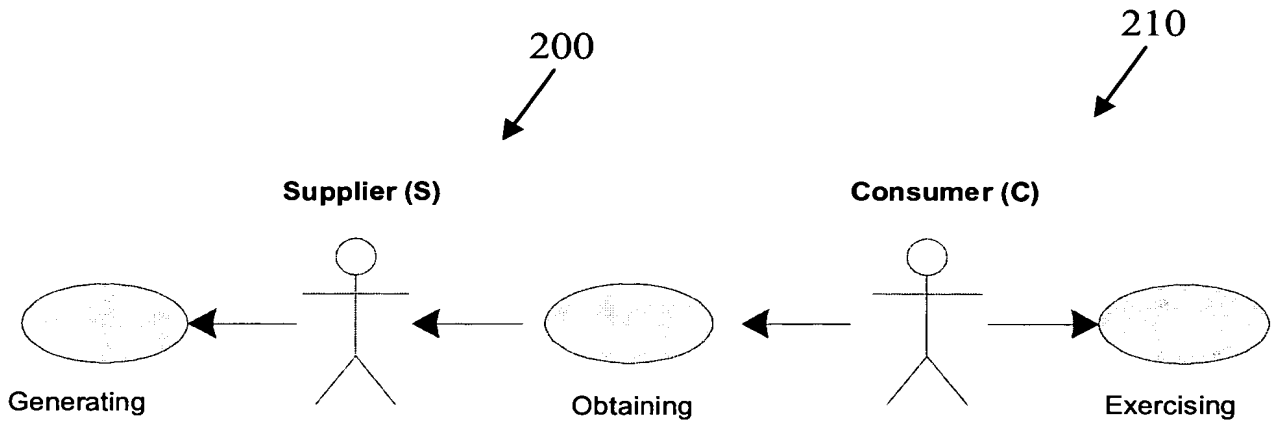


Fig. 3(b)

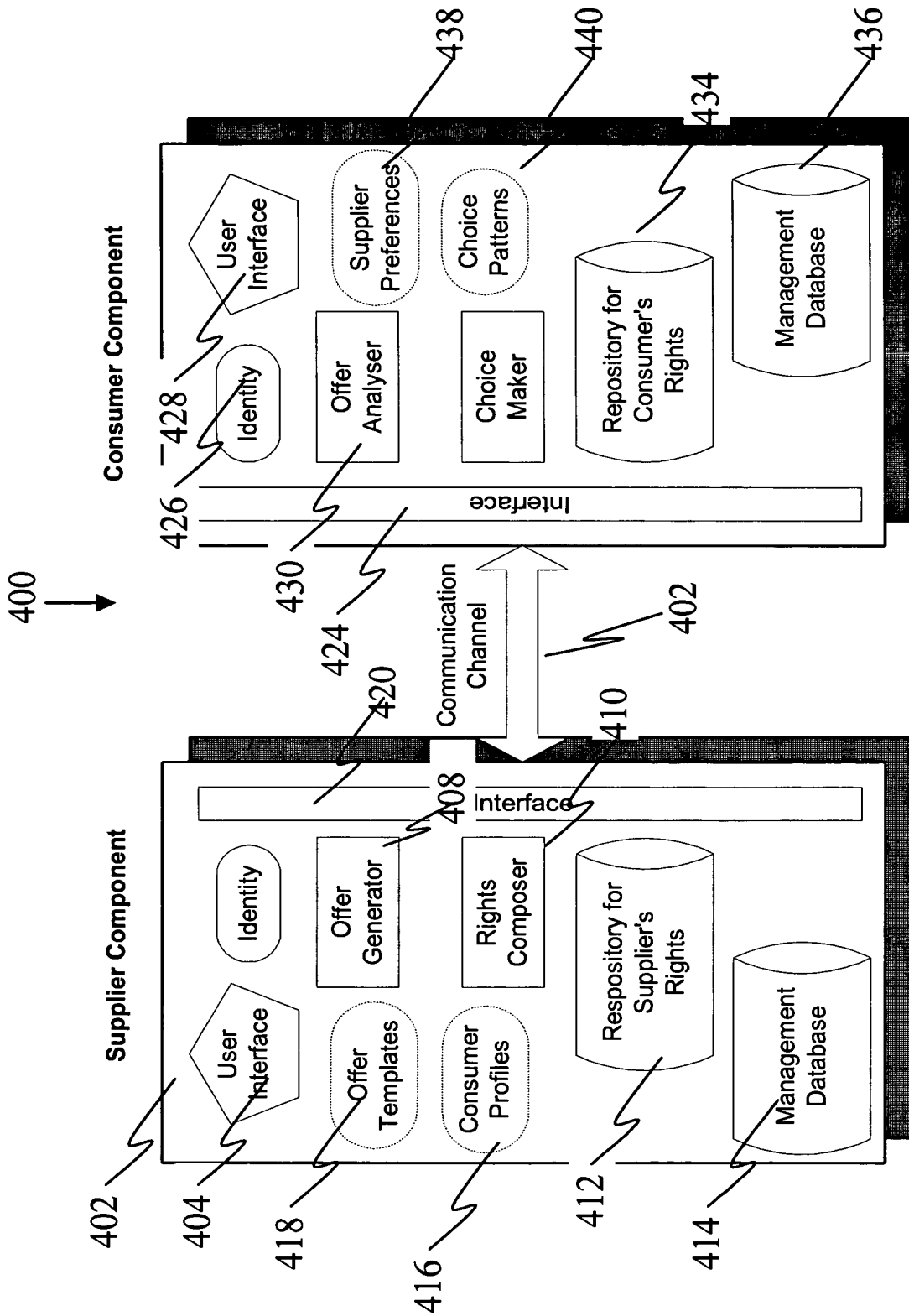


Fig. 4

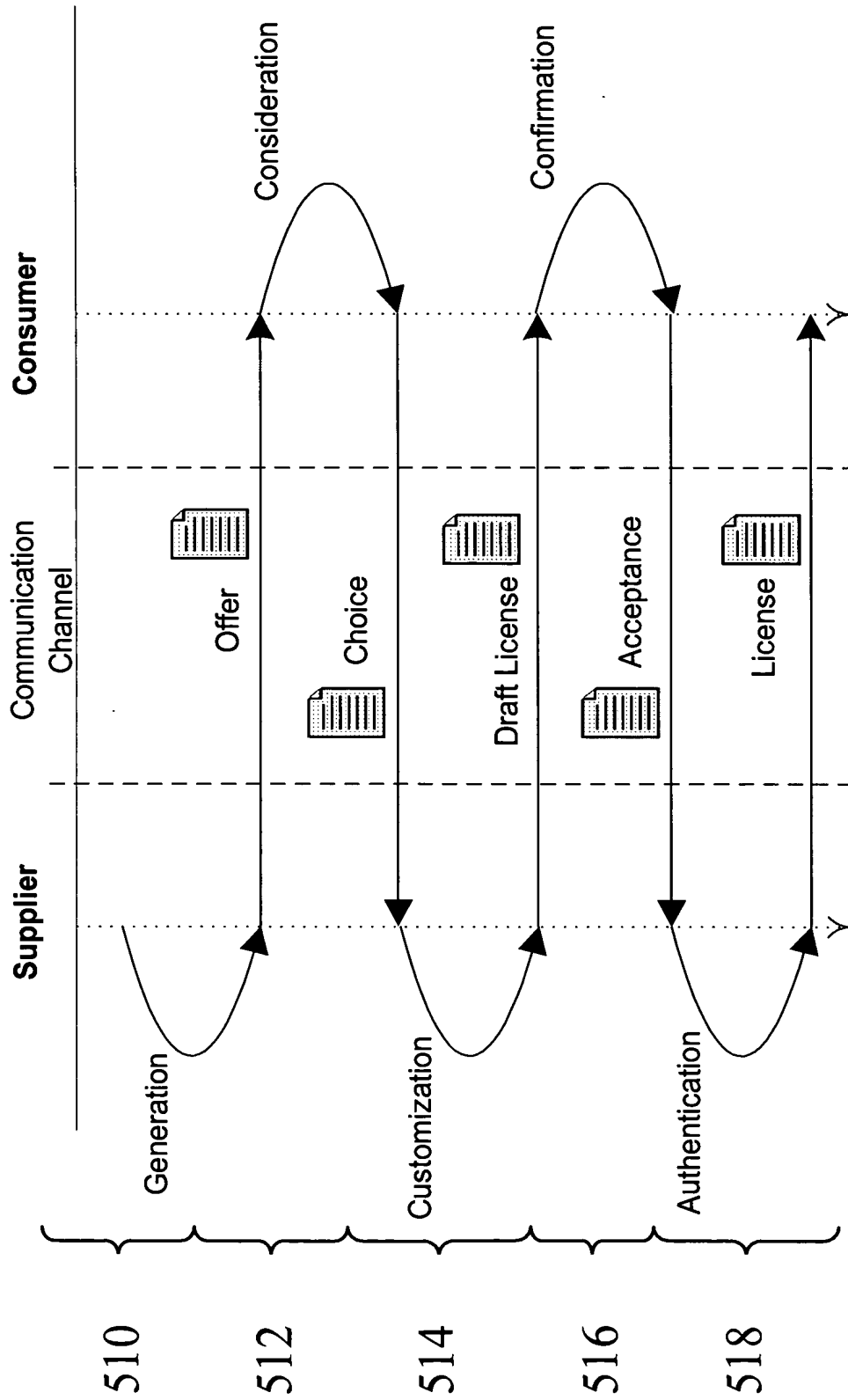


Fig. 5(a)

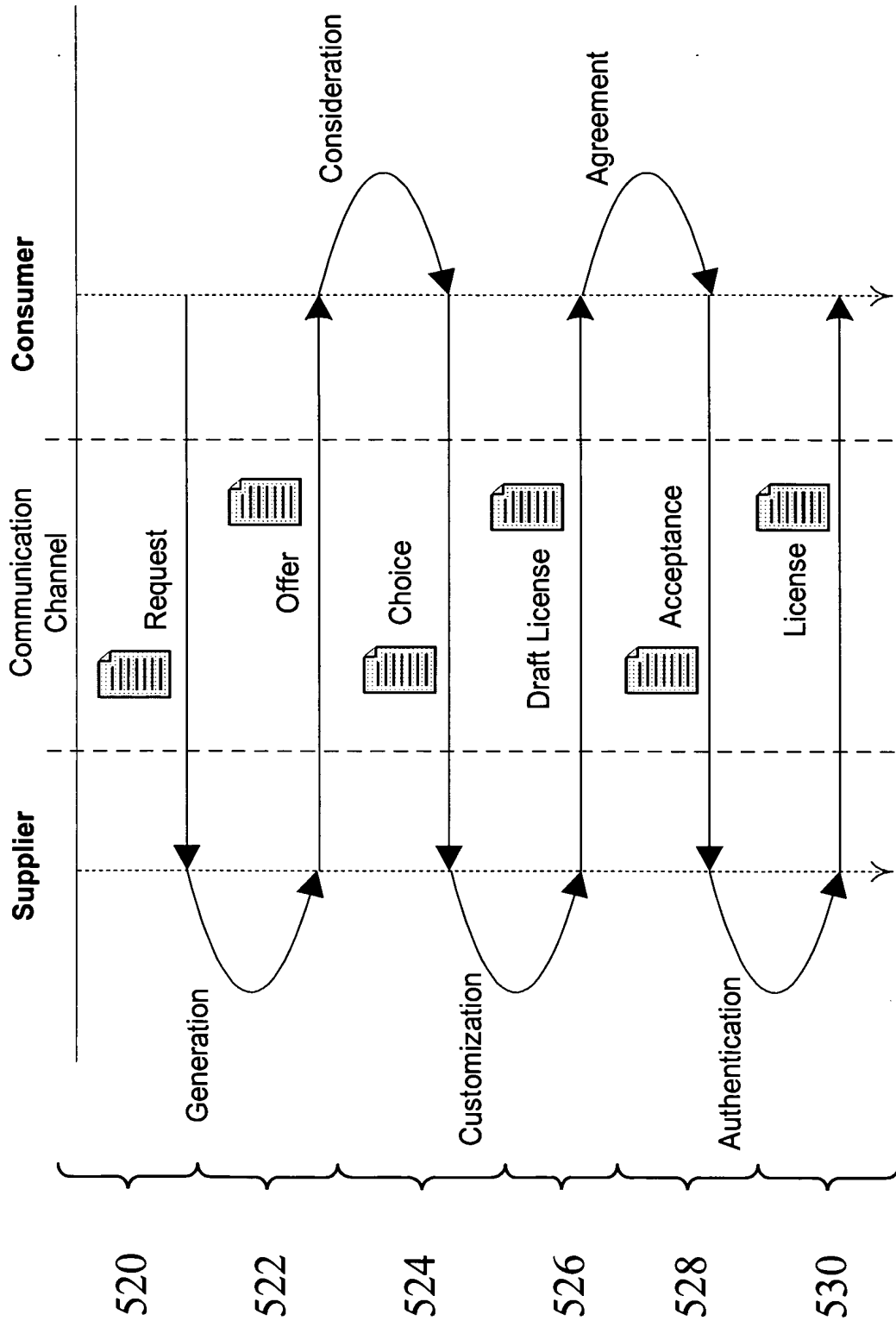


Fig. 5(b)

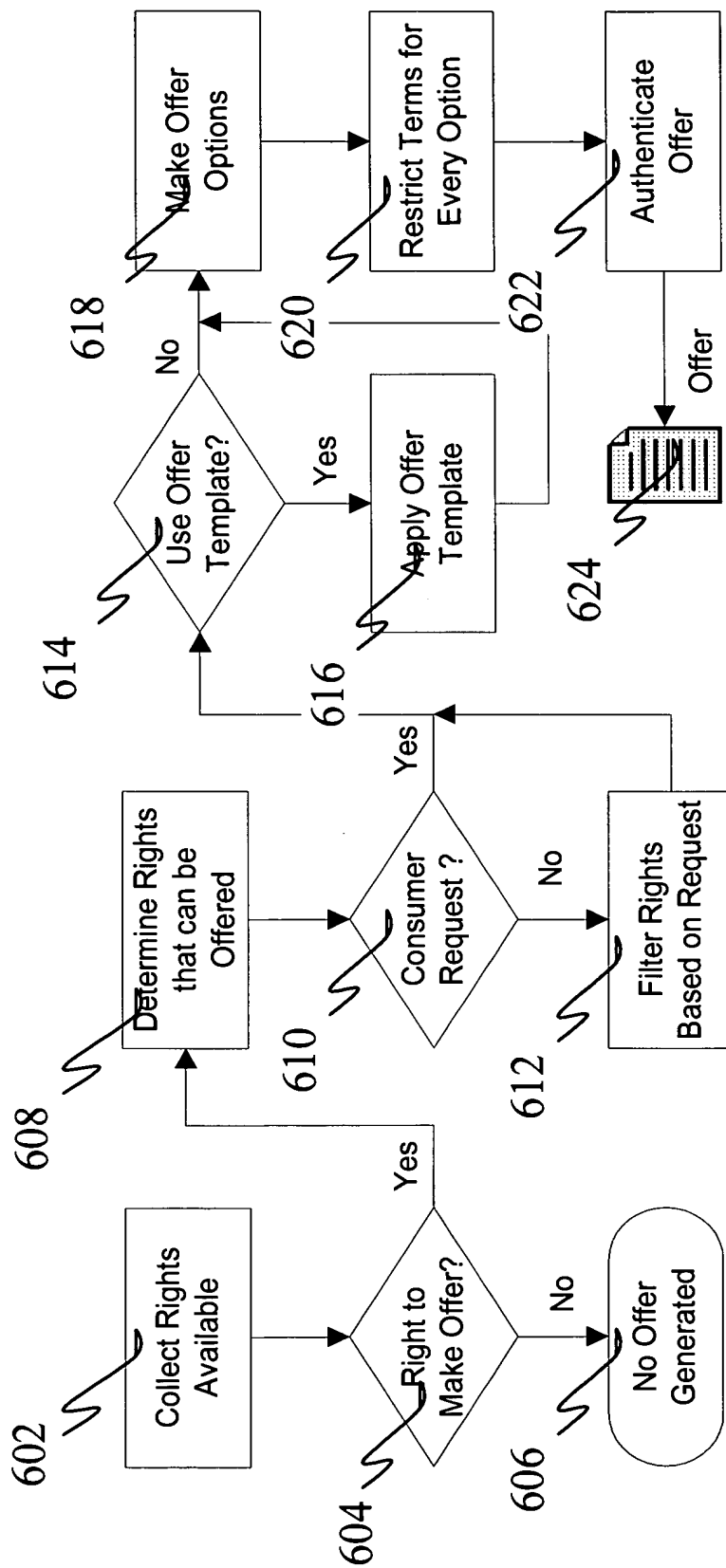


Fig. 6

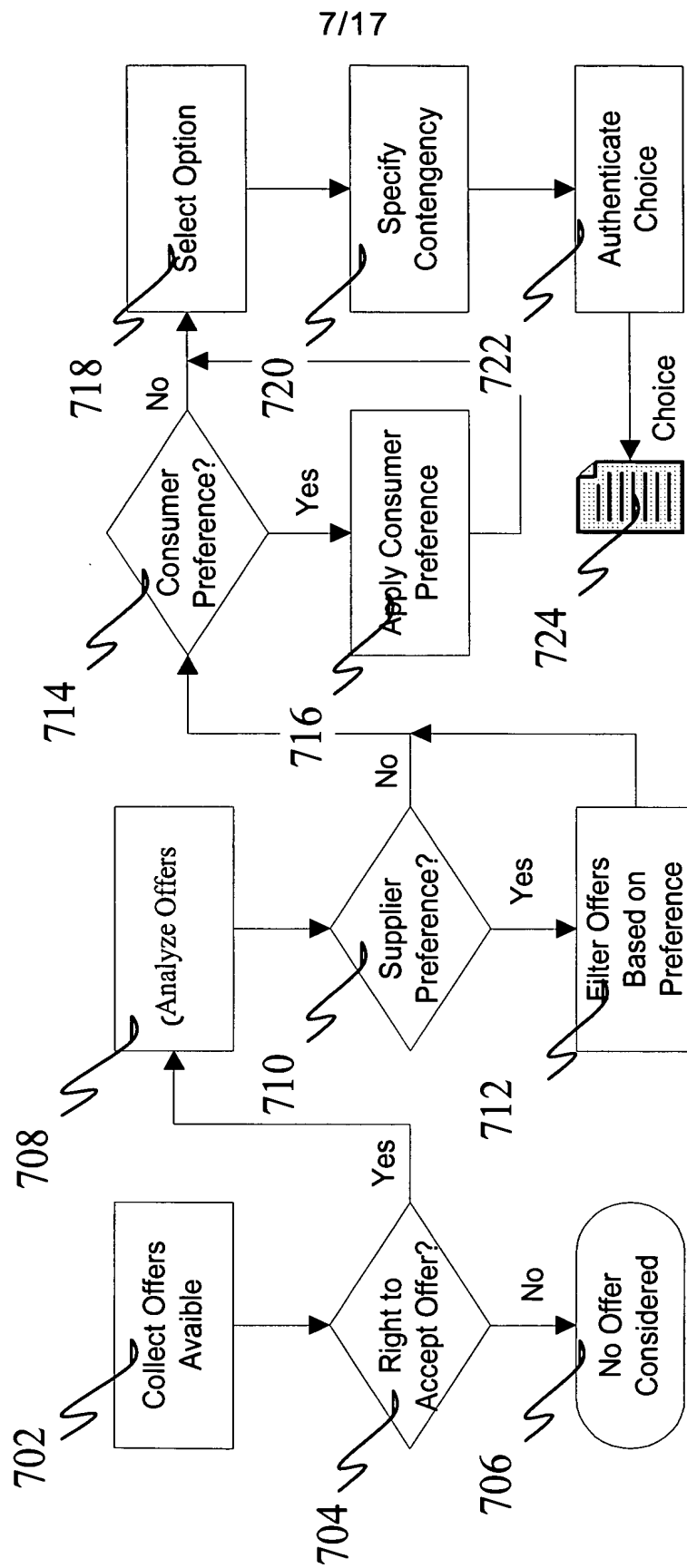


Fig. 7

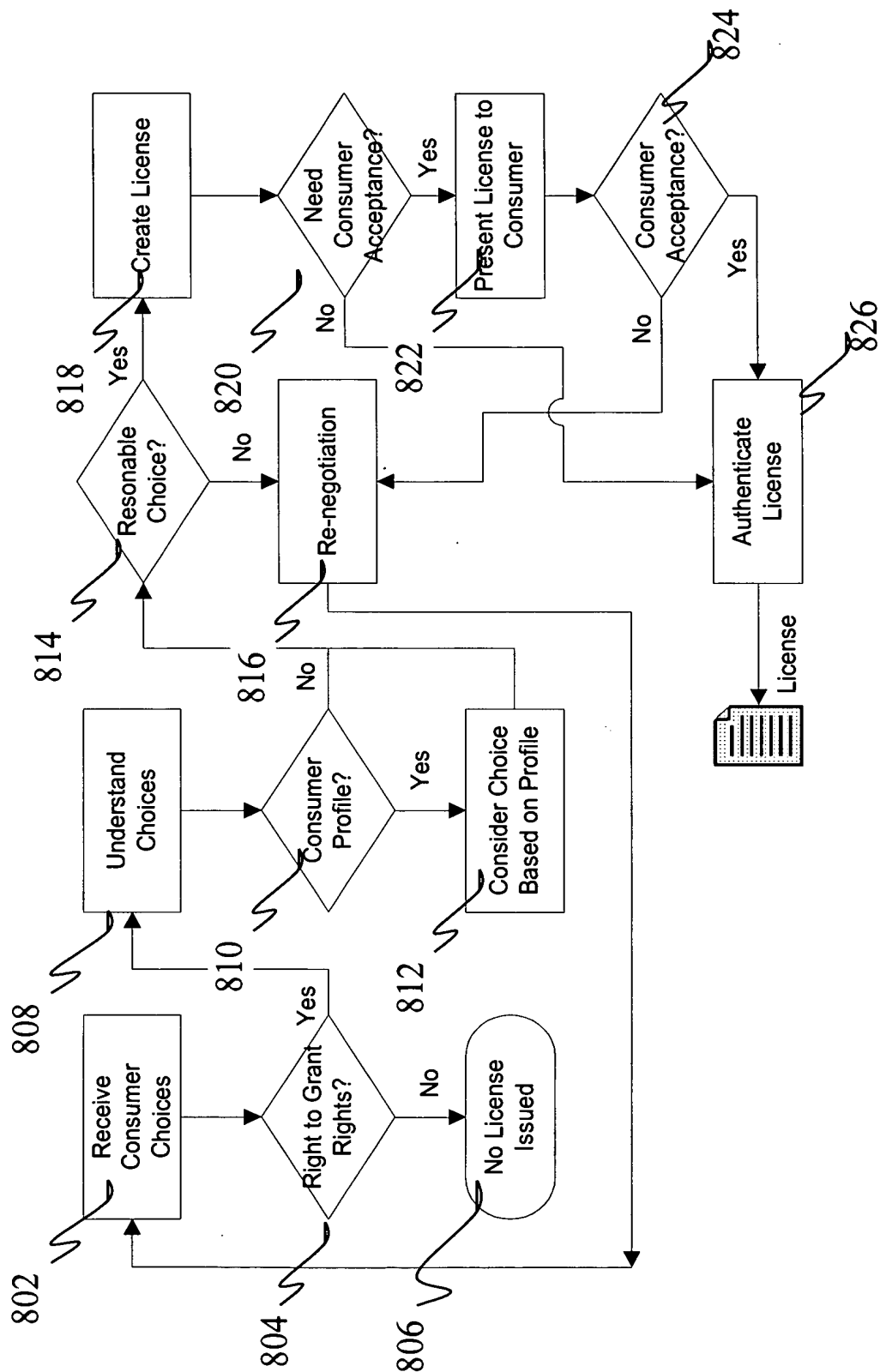


Fig. 8

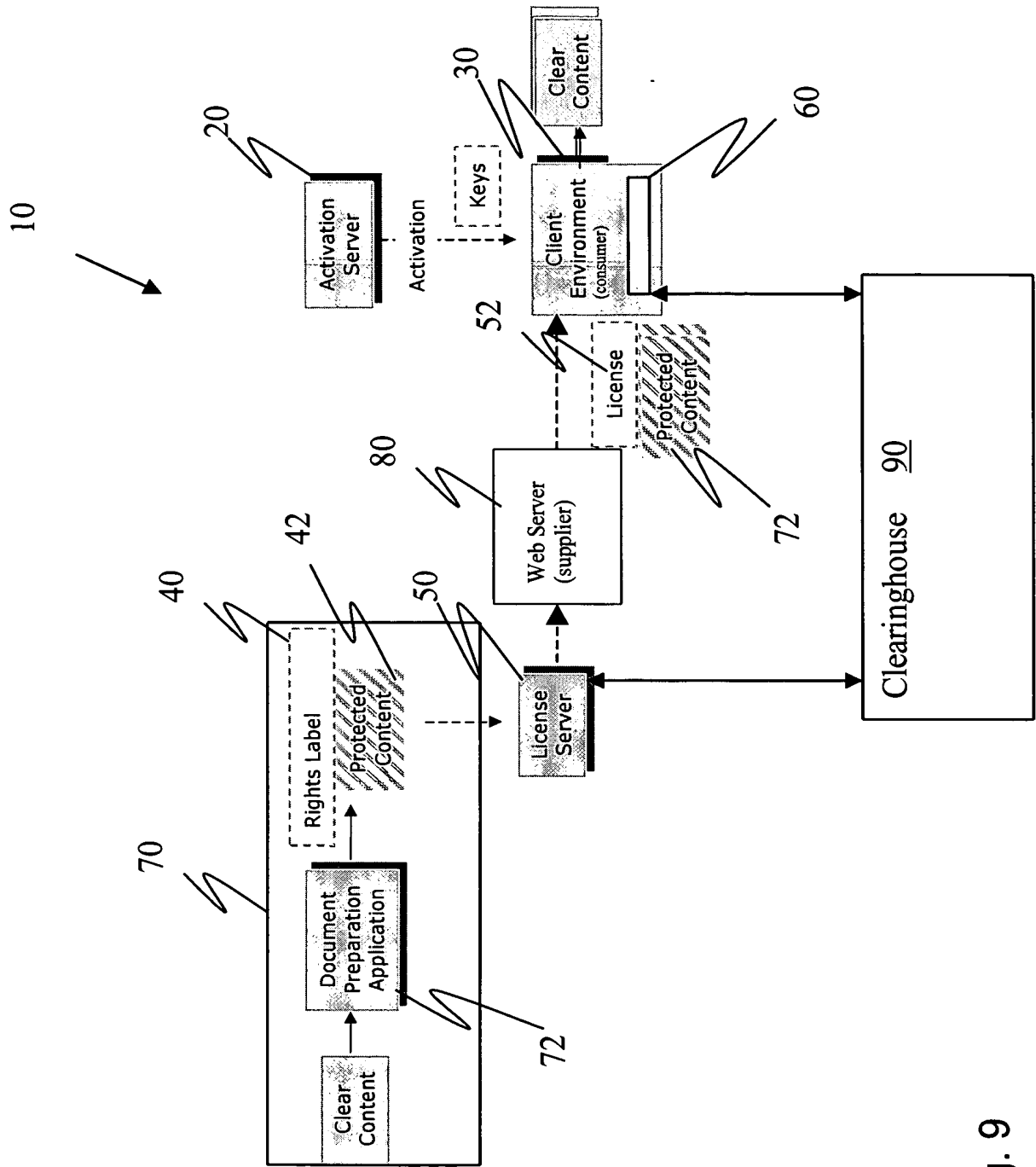


Fig. 9

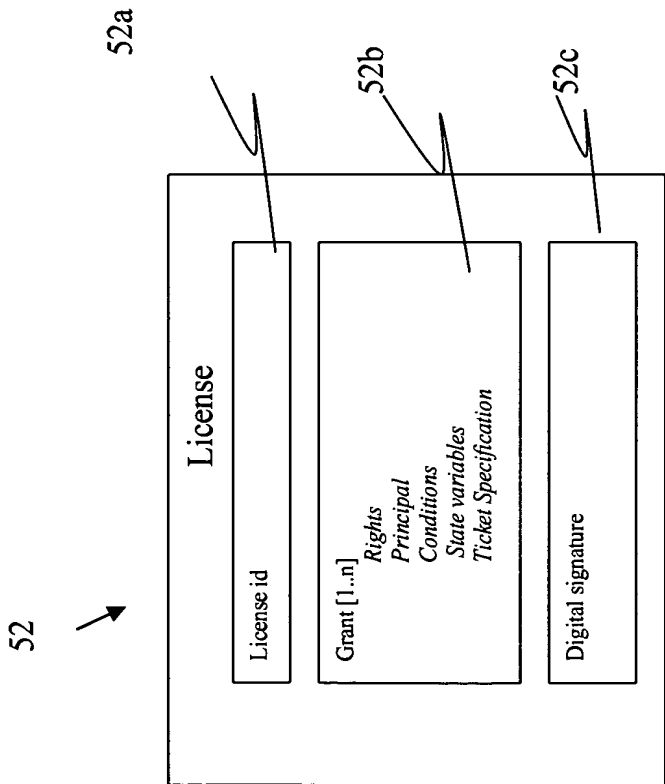
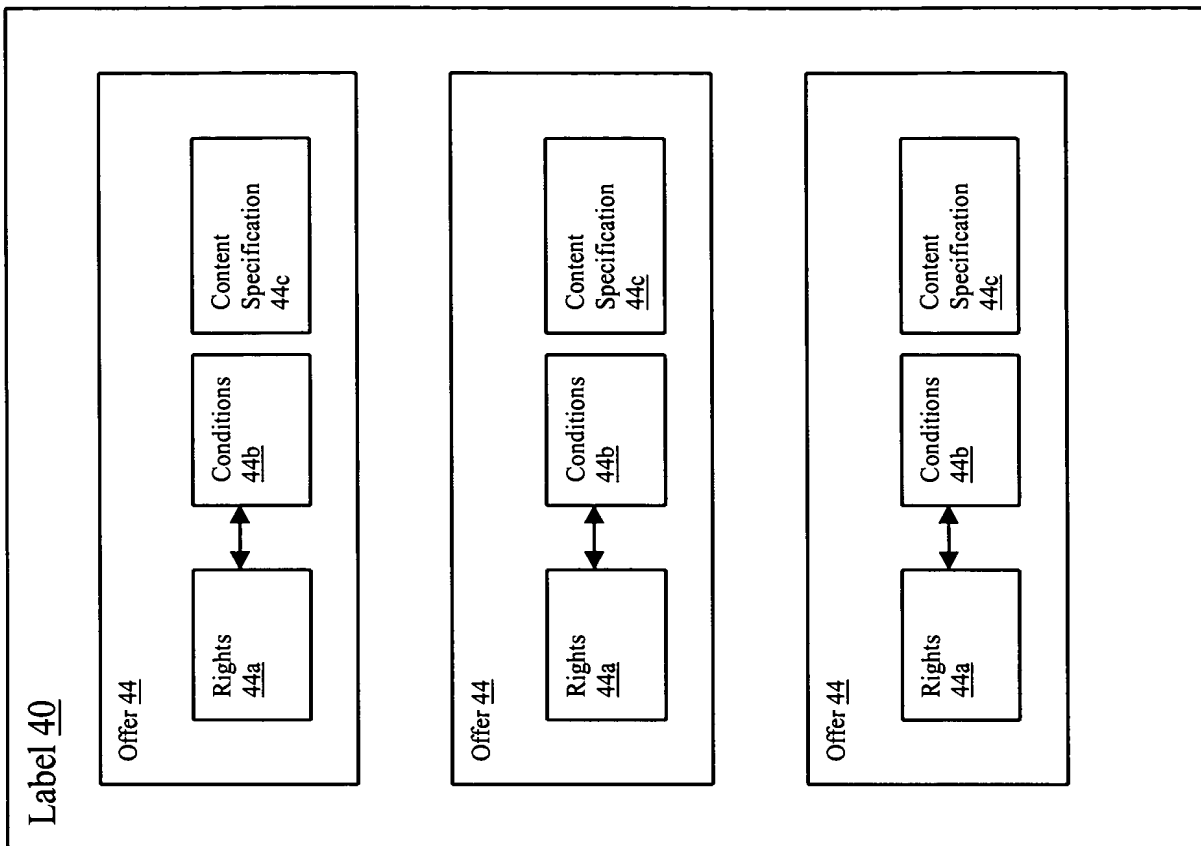


Fig. 10

Fig. 11

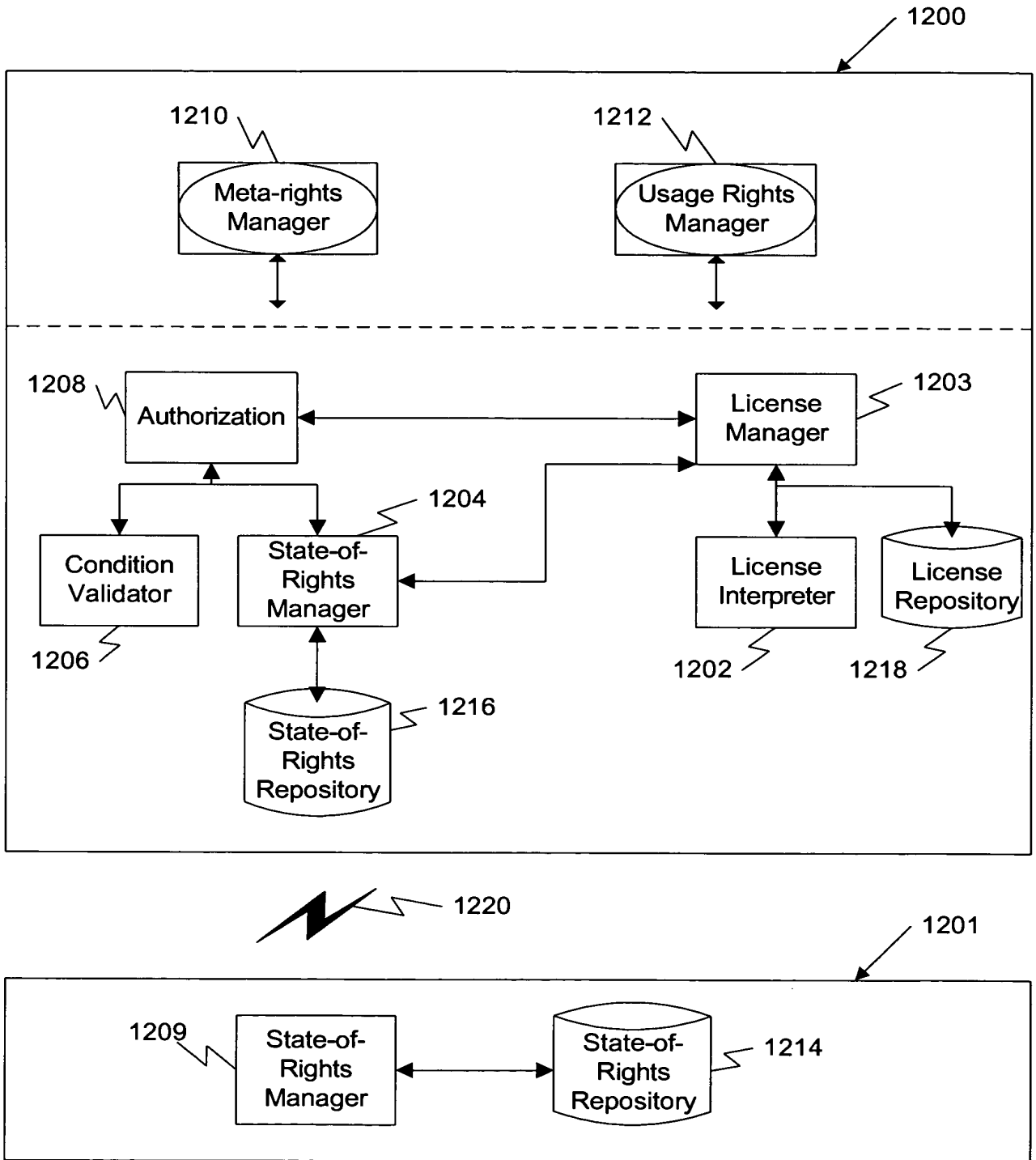


Fig. 12

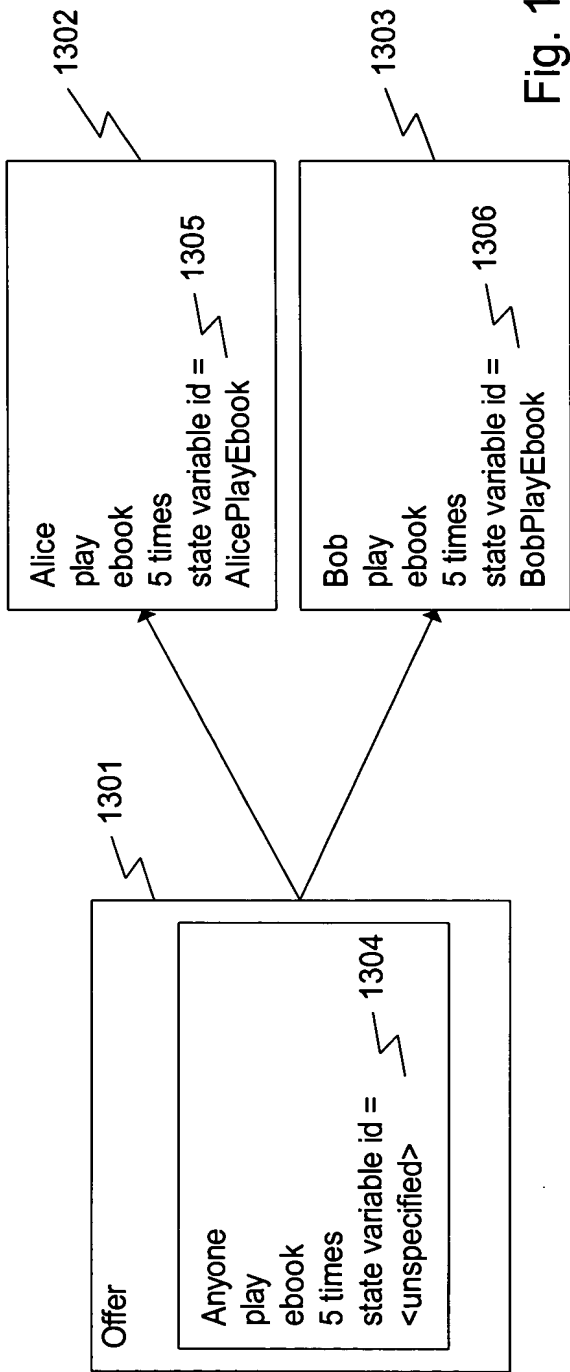


Fig. 13

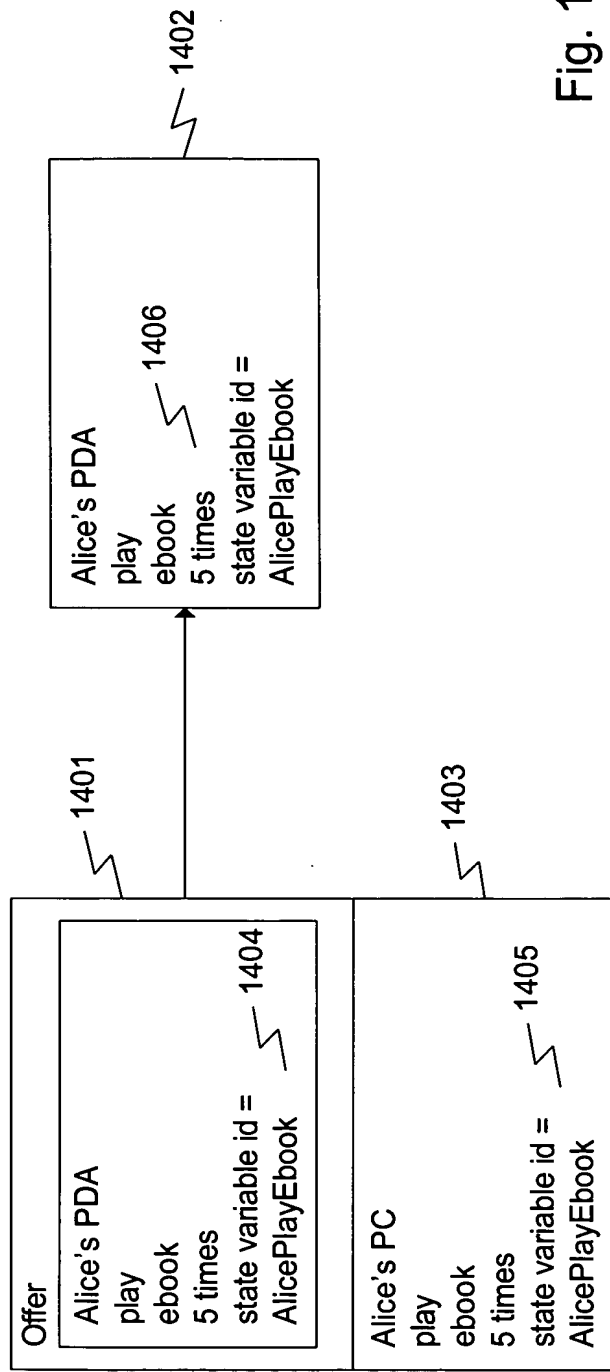


Fig. 14

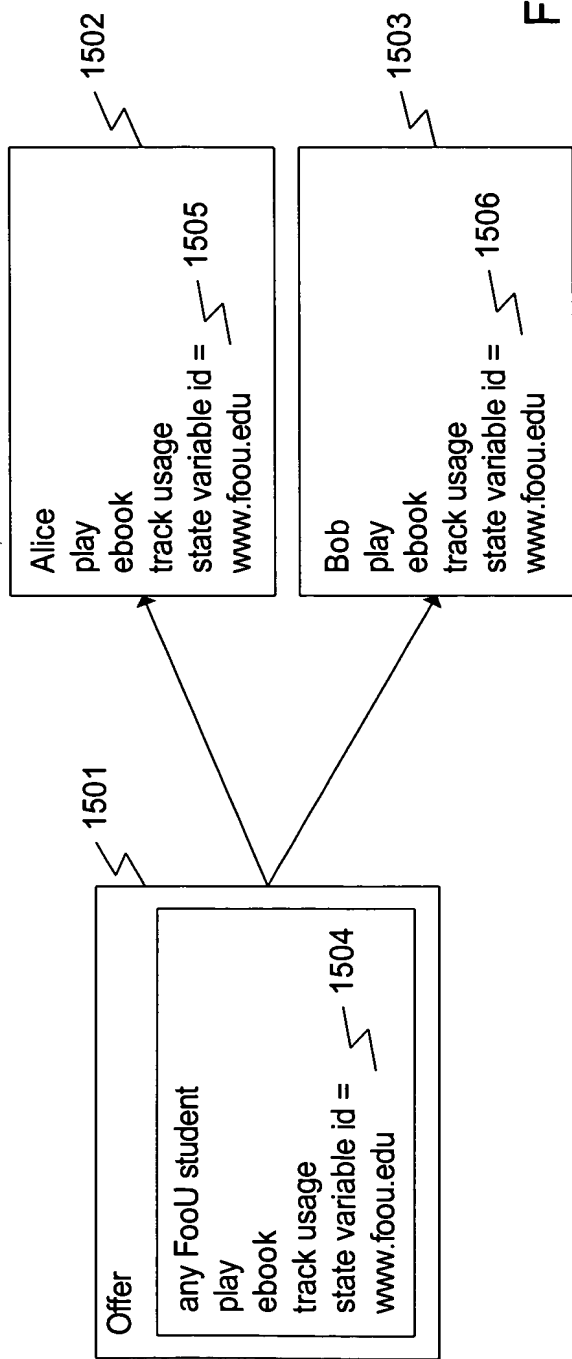


Fig. 15

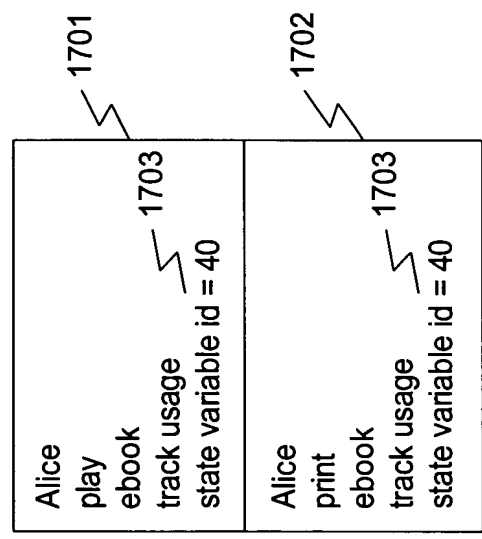


Fig. 17

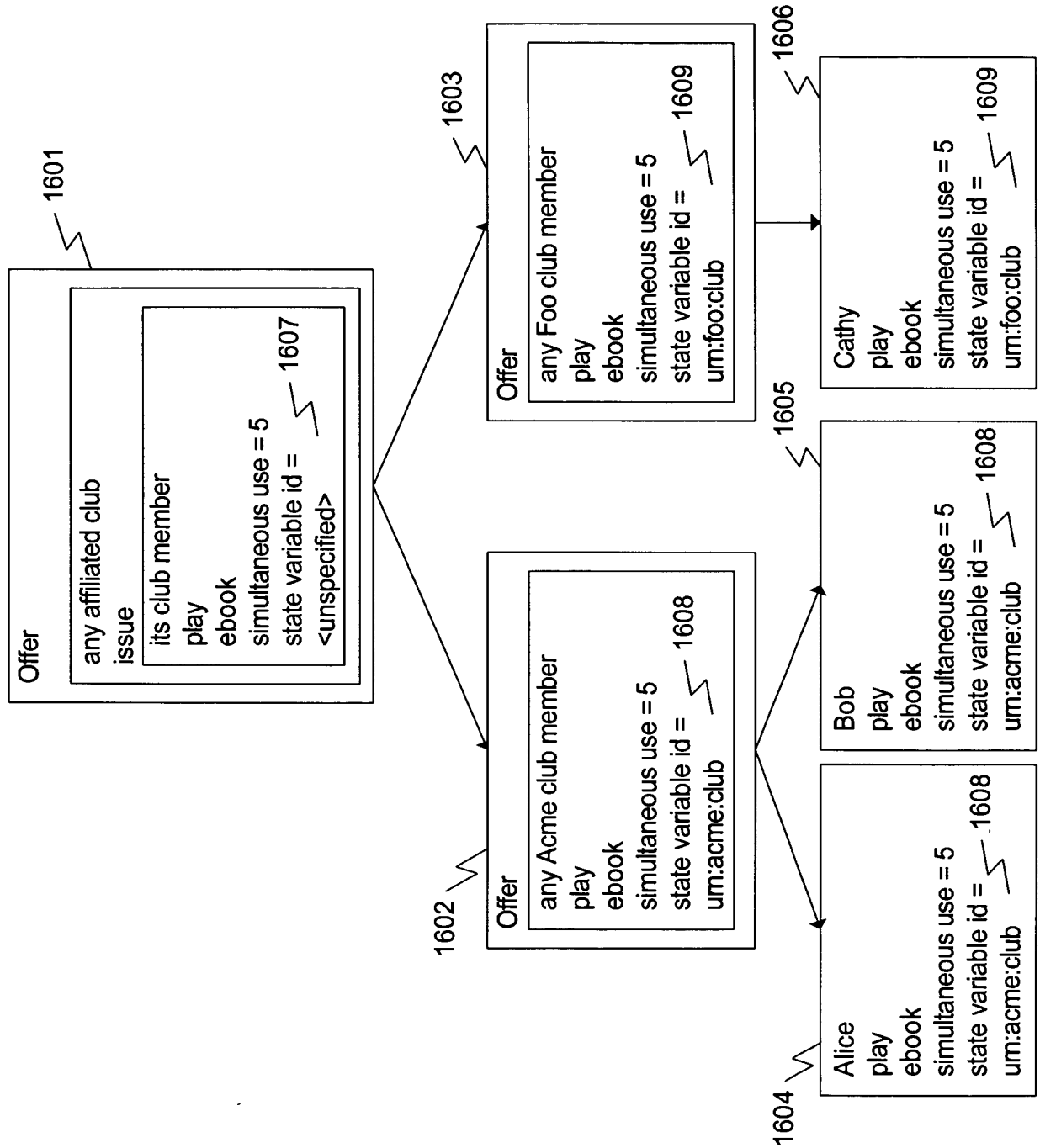


Fig. 16

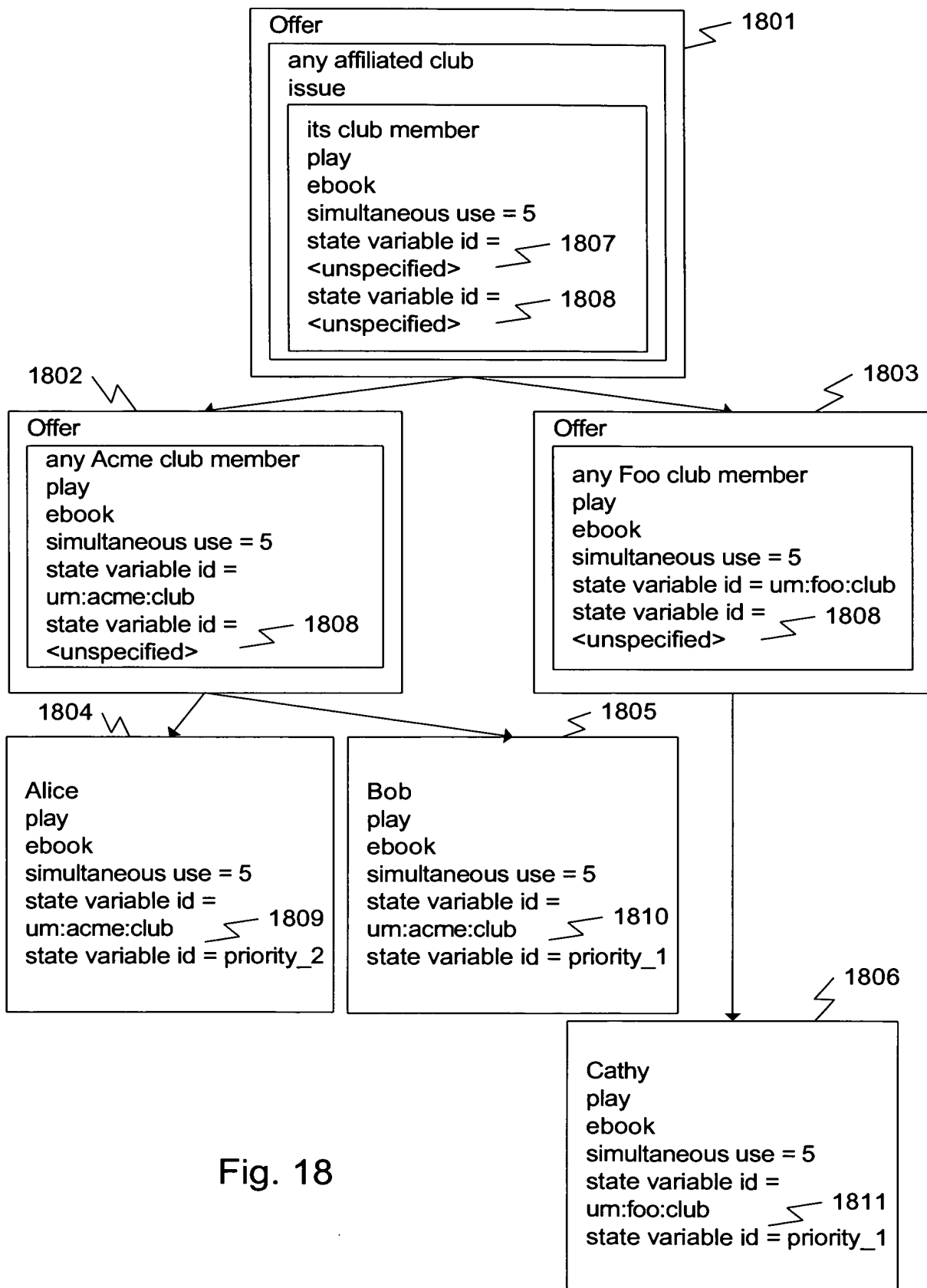


Fig. 18

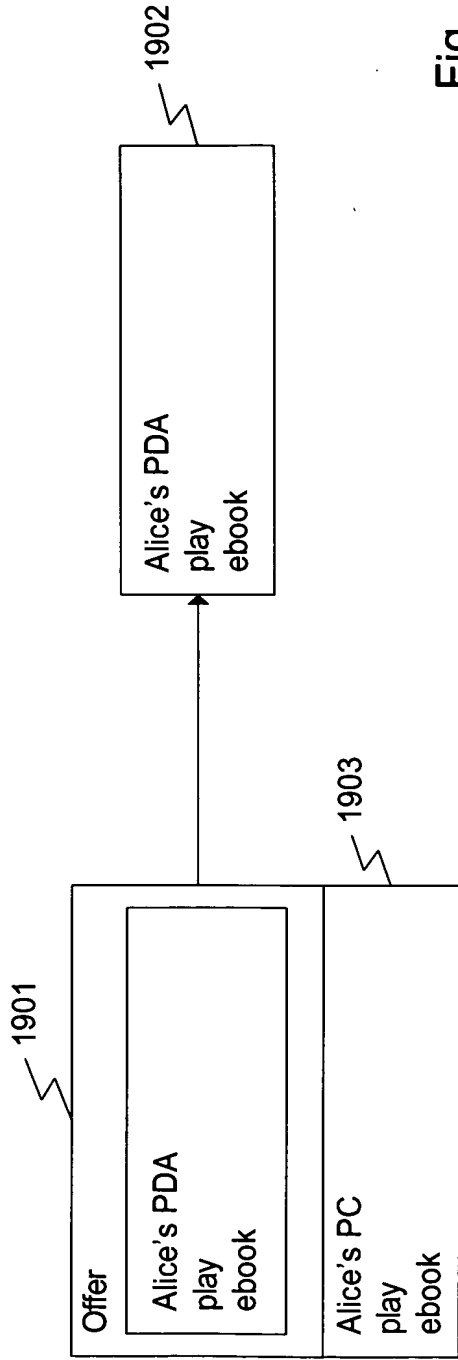


Fig. 19

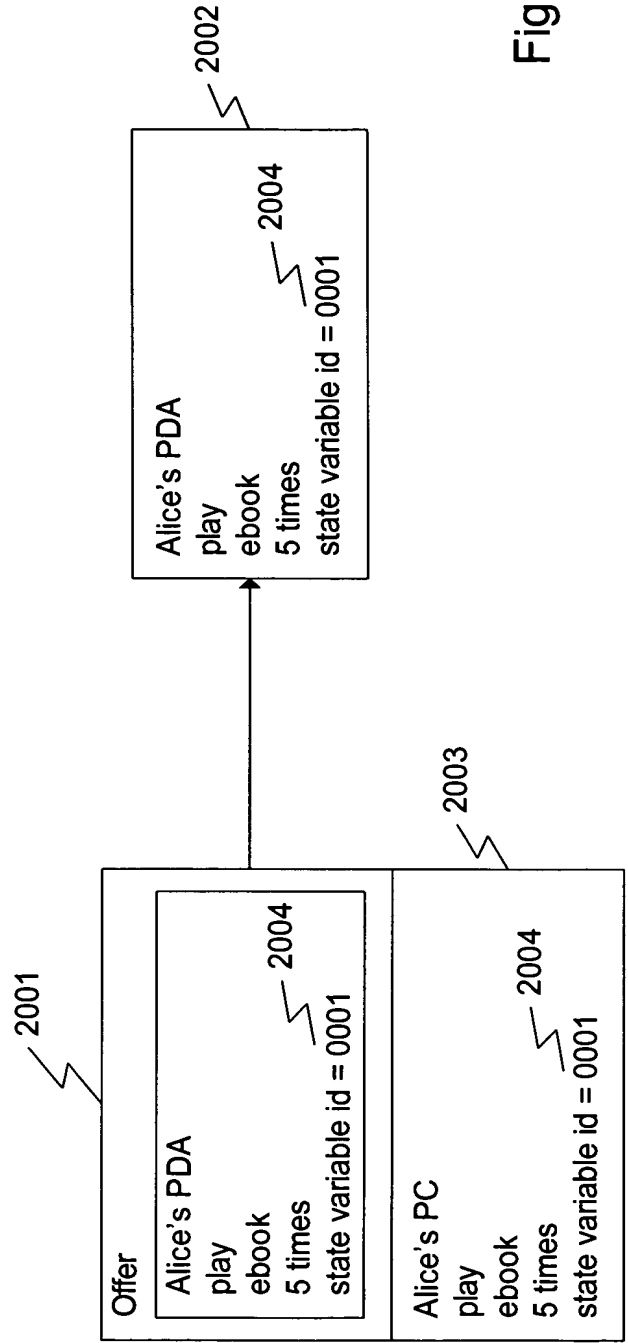


Fig. 20

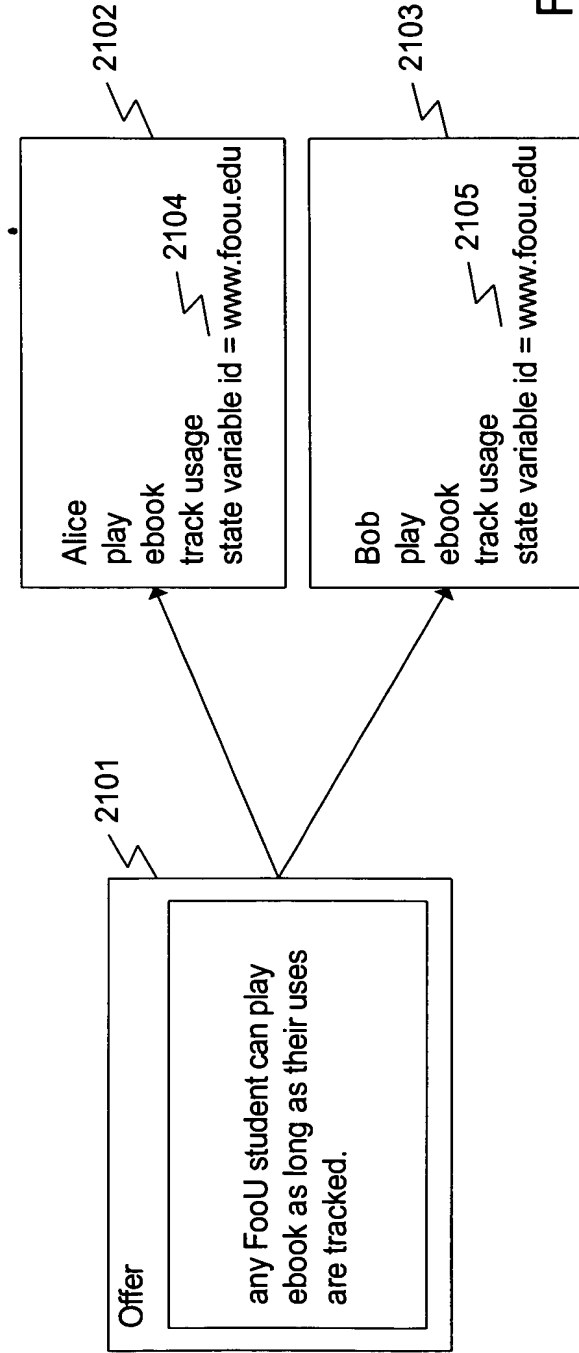


Fig. 21

Please type a plus sign (+) inside this box → [+]

PTO/SB/01 (12-97)
 Approved for use through 9/30/00. OMB 0651-0032
 Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OBM control number.

DECLARATION FOR UTILITY OR DESIGN PATENT APPLICATION (37 CFR 1.63) <input type="checkbox"/> Declaration Submitted With Initial Filing <input checked="" type="checkbox"/> Declaration Submitted after Initial Filing (surcharge (37 CFR 1.16(d)) required)	Attorney Docket Number	111325-104
	First Named Inventor	Xin WANG et al
	COMPLETE IF KNOWN	
	Application Number	10/162,212
	Filing Date	June 5, 2002
	Group Art Unit	3621
	Examiner Name	Not Yet Assigned

10956070
1987 U.S. PTO



As a below named inventor, I hereby declare that:
 My residence, post office address, and citizenship are as stated below next to my name.
 I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

RIGHTS OFFERING AND GRANTING _____
(Title of the Invention)

the specification of which
 is attached hereto
 OR
 was filed on (MM/DD/YYYY) June 5, 2002 As United States Application Number or PCT International Application Number 10/162,212
 And was amended on (MM/DD/YYYY) _____ (If applicable).
 I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment specifically referred to above.
 I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR 1.56.

I hereby claim foreign priority benefits under 35 U.S.C. 119(a)-(d) or 365(b) of any foreign application(s) for patent or inventor's certificate, or 365(a) of any PCT international application which designated at least one country other than the United States of America, listed below and have also identified below, by checking the box, any foreign application for patent or inventor's certificate, or of any PCT international application having a filing date before that of the application on which priority is claimed.

Prior Foreign Application Number(s)	Country	Foreign Filing Date (MM/DD/YYYY)	Priority Not Claimed	Certified Copy Attached	
				YES	No
			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Additional foreign application numbers are listed on a supplemental priority data sheet PTO/SB/02B attached hereto:
 I hereby claim the benefit under 35 U.S.C. 119(e) of any United States provisional application(s) listed below.

Application Number(s)	Filing Date (MM/DD/YYYY)	<input type="checkbox"/> Additional provisional application Numbers are listed on a supplemental priority data sheet PTO/SB/02B attached hereto.
60/296,113 60/331,625 60/331,624	June 7, 2001 November 20, 2001 November 20, 2001	

Burden Hour Statement: This form is estimated to take 0.4 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissions for Patents, Washington, DC 20231.

Please type a plus sign (+) inside this box → [+]

PTO/SB/01 (12-97)
Approved for use through 9/30/00. OMB 0651-0032
Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OBM control number.

DECLARATION – Utility or Design Patent Application

I hereby claim the benefit under 35 U.S.C. 120 of any United States application(s), or 365 of any PT international application designating the United States of America, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of 35 U.S.C. 112, I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application.

U.S. Parent Application or PCT Parent Number	Parent Filing Date (MM/DD/YYYY)	Parent Patent Number (if applicable)
This Application is a CIP of US Serial No. 09/867,745	May 31, 2001	

Additional U.S. or PCT international application are listed on a supplemental priority date sheet PTO/SB/02B attached hereto.

As a named inventor, I hereby appoint the following registered practitioner(s) to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith: Customer Number 22204
OR
 Registered practitioner(s) name/registration number listed below.

Name	Registration Number	Name	Registration Number
Stuart J. Friedman	24,312	Eric J. Robinson	38,285
Charles M. Leedom, Jr.	26,477	Daniel S. Song	43,143
David S. Safran	27,997	Marc S. Kaufman	35,212
Thomas W. Cole	28,290	Corinne R. Gorski	34,339
Donald R. Studebaker	32,815	Jason H. Vick	45,285
Jeffrey L. Costellia	35,483	Luan C. Do	38,434
Tim L. Brackett, Jr.	36,092		

Direct all correspondence to: Customer Number 22204

Name: Marc S. Kaufman

Firm: NIXON PEABODY LLP

Address: 8180 Greensboro Drive, Suite 800

City: McLean State: VA ZIP: 22102

Country: United States Telephone: (703) 770-9300 FAX: (703) 9400

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. 1001 and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Name of Sole or First Inventor: A petition has been filed for this unsigned inventor.

Given Name (first and middle [if any])	Family Name or Surname
Xin	WANG

Inventor's Signature: Date: 8/15/2002

Mailing Address (Street or P.O.-Box): 3720 Emerald Street #V2

City: Torrance State: CA ZIP: 90503 Country: USA

Residence: City: Torrance State: Country:

Citizenship: USA

Additional inventors are being named on the _____ Supplemental Additional Inventor(s) sheet(s) PTO/SB/02A attached hereto.

Please type a plus sign (+) inside this box → [+]

PTO/SB/02A (10-00)

Approved for use through 10/31/2002. OMB 0651-0032
Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OBM control number.

DECLARATION	ADDITIONAL INVENTOR(S) Supplemental Sheet Page ___ of ___
--------------------	--

Name of Additional Joint Inventor, if any:	<input type="checkbox"/> A petition has been filed for this unsigned inventor		
Given Name Bijan	Family Name Or Surname TADAYON		
Inventor's Signature B. Tadayon	Date AUG-7-02		
Mailing Address (Street or P.O. Box): 20920 Scottsbury Dr.			
City: Germantown	State: MD	ZIP: 20876	Country: USA
Residence: City: Germantown	State:	Country:	
Citizenship: USA			
Name of Additional Joint Inventor, if any:	<input type="checkbox"/> A petition has been filed for this unsigned inventor		
Given Name	Family Name Or Surname		
Inventor's Signature	Date		
Mailing Address (Street or P.O. Box):			
City:	State:	ZIP:	Country:
Residence: City:	State:	Country:	
Citizenship:			
Name of Additional Joint Inventor, if any:	<input type="checkbox"/> A petition has been filed for this unsigned inventor		
Given Name	Family Name Or Surname		
Inventor's Signature	Date		
Mailing Address (Street or P.O. Box):			
City:	State:	ZIP:	Country:
Residence: City:	State:	Country:	
Citizenship:			

Burden Hour Statement: This form is estimated to take 21 minutes to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

[Page 3 of 3]

Please type a plus sign (+) inside this box → [+]

Approved for use through 10/31/2002. OMB 0651-0032
PTO/SB/02A (10-00)

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- BLACK BORDERS
- IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT OR DRAWING
- BLURRED OR ILLEGIBLE TEXT OR DRAWING
- SKEWED/SLANTED IMAGES
- COLOR OR BLACK AND WHITE PHOTOGRAPHS
- GRAY SCALE DOCUMENTS
- LINES OR MARKS ON ORIGINAL DOCUMENT
- REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

PATENT APPLICATION FEE DETERMINATION RECORD
Effective October 1, 2004

Application or Docket Number

10956070

CLAIMS AS FILED - PART I

	(Column 1)	(Column 2)
TOTAL CLAIMS	39	
FOR	NUMBER FILED	NUMBER EXTRA
TOTAL CHARGEABLE CLAIMS	39 minus 20 = *	19
INDEPENDENT CLAIMS	3 minus 3 = *	0
MULTIPLE DEPENDENT CLAIM PRESENT <input type="checkbox"/>		

SMALL ENTITY TYPE

OR OTHER THAN SMALL ENTITY

RATE	FEE		RATE	FEE
BASIC FEE	395.00	OR	BASIC FEE	790.00
X\$ 9=		OR	X\$18=	392.00
X44=		OR	X88=	
+150=		OR	+300=	
TOTAL		OR	TOTAL	1132.00

* If the difference in column 1 is less than zero, enter "0" in column 2

CLAIMS AS AMENDED - PART II

	(Column 1)	(Column 2)	(Column 3)
AMENDMENT A	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total *	Minus **	=
	Independent *	Minus ***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>			

SMALL ENTITY OR

OTHER THAN SMALL ENTITY

RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
X\$ 9=		OR	X\$18=	
X44=		OR	X88=	
+150=		OR	+300=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

	(Column 1)	(Column 2)	(Column 3)
AMENDMENT B	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total *	Minus **	=
	Independent *	Minus ***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>			

RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
X\$ 9=		OR	X\$18=	
X44=		OR	X88=	
+150=		OR	+300=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

	(Column 1)	(Column 2)	(Column 3)
AMENDMENT C	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total *	Minus **	=
	Independent *	Minus ***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>			

RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
X\$ 9=		OR	X\$18=	
X44=		OR	X88=	
+150=		OR	+300=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20."

***If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3."

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

PATENT APPLICATION SERIAL NO. _____

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

10/06/2004 ANABI1 00000075 192380 10956070

01 FC:1001	790.00 DA
02 FC:1202	342.00 DA

PTO-1556
(5/87)

APPLICATION DATA SHEET

Electronic Version 0.0.11

Stylesheet Version: 1.0

Attorney Docket Number: 111325-23500

Publication Filing Type: new-utility

Application Type: utility

Title of Invention: SYSTEM AND METHOD FOR RIGHTS OFFERING AND GRANTING
USING SHARED STATE VARIABLES

Legal Representative:

Attorney or Agent: Marc S. Kaufman

Registration Number: 35212

Attorney or Agent: Carlos R. Villamar

Registration Number: 43224

Customer Number Correspondence Address: 22204

22204

Continuity Data:

This application is a continuation in part of 10/162,212 2002-06-05 which is a continuation-in-part application of application serial no. 09/867,745 filed on May 31, 2001 and which claims benefit from U.S. Provisional Application Serial No. 60/296,113 filed on June 7, 2001, U.S. Provisional Serial No. 60/331,625 filed on November 20, 2001 and U.S. Provisional Application Serial No. 60/331,624 filed on November 20, 2001

INVENTOR(s):

Primary Citizenship: United States

Given Name: Mai

Family Name: Nguyen

Residence City: Buena Park

Residence State: CA

Residence Country: US

Address: 5611 Cambridge Avenue
Buena Park CA, 96021 US

Primary Citizenship: United States
Given Name: Xin
Family Name: Wang
Residence City: Torrance
Residence State: CA
Residence Country: US
Address: 3720 Emerald Street, #V2
Torrance CA, US

Primary Citizenship: United States
Given Name: Eddie
Middle Name: J.
Family Name: Chen
Residence City: Rancho Palos Verdes
Residence State: CA
Residence Country: US
Address: 6796 Vallon Drive
Rancho Palos Verdes CA, 90275 US

Primary Citizenship: United States
Given Name: Bijan
Family Name: Tadayon
Residence City: Germantown
Residence State: MD
Residence Country: US
Address: 20920 Scottsbury Drive
Germantown Maryland, 20876 US

100404
13281 U.S. PTO

Approved for use through 10/31/2002. OMB 0651-0032
PTO/SB/05 (03-01)

UTILITY PATENT APPLICATION TRANSMITTAL <i>(Only for new nonprovisional applications under 37 CFR 1.53(b))</i>	Attorney Docket No. 111325-235000	
	First Inventor Mai NGUYEN, et al.	
	Title	SYSTEM AND METHOD FOR RIGHTS OFFERING AND GRANTING USING SHARED STATE VARIABLES
	Express Mail Label No.	

APPLICATION ELEMENTS See MPEP chapter 600 concerning utility patent application contents.	ADDRESS TO: Commissioner for Patents Box Patent Application Washington, DC 20231
---	--

1. <input checked="" type="checkbox"/> Fee Transmittal Form (e.g., PTO/SB/17) <i>(Submit an original and a duplicate for fee processing)</i> 2. <input type="checkbox"/> Applicant claims small entity status. See 37 CFR 1.27. 3. <input checked="" type="checkbox"/> Specification [Total Pages 43] <i>(preferred arrangement set forth below)</i> - Descriptive title of the invention - Cross Reference to Related Applications <i>(if applicable)</i> - Statement Regarding Fed sponsored R & D <i>(if applicable)</i> - Reference to sequence listing, a table, or a computer program listing appendix <i>(if applicable)</i> - Background of the Invention - Brief Summary of the Invention - Brief Description of the Drawings <i>(if filed)</i> - Detailed Description - Claim(s) - Abstract of the Disclosure 4. <input checked="" type="checkbox"/> Drawing(s) (35 U.S.C. 113) 23 Figures [Total Sheets 17] 5. Oath or Declaration [Total Pages <input type="checkbox"/>] a. <input type="checkbox"/> Newly executed (original or copy) b. <input checked="" type="checkbox"/> Copy from a prior application (37 CFR 1.63(d)) <i>(for continuation/divisional with Box 18 completed)</i> i. <input type="checkbox"/> DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b) 6. <input checked="" type="checkbox"/> Application Data Sheet. See 37 CFR 1.76	7. <input type="checkbox"/> CD-ROM or CD-R in duplicate, large table or Computer Program (Appendix) 8. Nucleotide and/or Amino Acid Sequence Submission <i>(if applicable, all necessary)</i> a. <input type="checkbox"/> Computer Readable Form (CRF) b. Specification Sequence Listing on: i. <input type="checkbox"/> CD-ROM or CD-R (2 copies; or ii. <input type="checkbox"/> paper c. <input type="checkbox"/> Statements verifying identity of above copies
--	--

ACCOMPANYING APPLICATION PARTS
9. <input type="checkbox"/> Assignment Papers (cover sheet & document(s)) 10. <input type="checkbox"/> 37 CFR 3.73(b) Statement <input type="checkbox"/> Power of Attorney <i>(when there is an assignee)</i> 11. <input type="checkbox"/> English Translation Document <i>(if applicable)</i> 12. <input type="checkbox"/> Information Disclosure Statement (IDS)/PTO-1449 <input type="checkbox"/> Copies of IDS Citations 13. <input type="checkbox"/> Preliminary Amendment 14. <input checked="" type="checkbox"/> Return Receipt Postcard (MPEP 503) <i>(Should be specifically itemized)</i> 15. <input type="checkbox"/> Certified Copy of Priority Document(s) <i>(if foreign priority is claimed)</i> 16. <input type="checkbox"/> Nonpublication request under 35 U.S.C. 122(b)(2)(B)(i). Applicant must attach form PTO/SB/35 or its equivalent. 17. <input type="checkbox"/> Other: _____

18. If a CONTINUING APPLICATION, check appropriate box, and supply the requisite information below and in a preliminary amendment, or in an Application Data Sheet under 37 CFR 1.76:

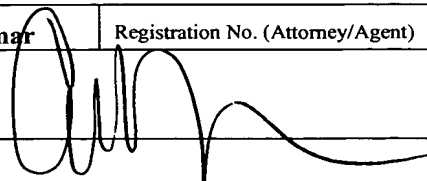
Continuation Continuation-in-part (CIP) of prior application No: 10/162,212

Prior application information: Examiner Not Yet Assigned Group / Art Unit: 3621

For CONTINUATION OR DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 5b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

19. CORRESPONDENCE ADDRESS	
<input checked="" type="checkbox"/> Customer Number or Bar Code Label	22204

Name			
Address			
City	State	Zip Code	
Country	Telephone	Fax	

Name (Print/Type)	Carlos R. Villamar	Registration No. (Attorney/Agent)	43,224
Signature		Date	October 4, 2004

100404

3281 U.S. PATENT OFFICE

FEE TRANSMITTAL FOR FY 2004

Patent fees are subject to annual revision.

Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT **\$1,132.00**

Complete if Known

Application Number	Not Yet Assigned
Filing Date	October 4, 2004
First Named Inventor	Mai NGUYEN, et al.
Examiner Name	Not Yet Assigned
Art Unit	Not Yet Assigned
Attorney Docket No.	111325-235000

METHOD OF PAYMENT (check all that apply)

Check Credit Card Money Order Other None

Deposit Account:

Deposit Account Number **19-2380**

Deposit Account Name **Nixon Peabody LLP**

The Commissioner is authorized to: (check all that apply)

Charge fee(s) indicated below Credit any overpayments

Charge any additional fee(s)

Charge fee(s) indicated below, except for the filing fee to the above-identified deposit account.

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity		Small Entity		Fee Description
Fee Code	Fee (\$)	Fee Code	Fee (\$)	
1051	130	2051	65	Surcharge - late filing fee or oath
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet
1053	130	1053	130	Non-English specification
1812	2,520	1812	2,520	For filing a request for <i>ex parte</i> reexamination
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action
1251	110	2251	55	Extension for reply within first month
1252	430	2252	215	Extension for reply within second month
1253	980	2253	490	Extension for reply within third month
1254	1,530	2254	765	Extension for reply within fourth month
1255	2,080	2255	1,040	Extension for reply within fifth month
1401	340	2401	170	Notice of Appeal
1402	340	2402	170	Filing a brief in support of an appeal
1403	340	2403	150	Request for oral hearing
1451	1,510	1451	1,510	Petition to institute a public use proceeding
1452	110	2452	55	Petition to revive - unavoidable
1453	1,370	2453	685	Petition to revive - unintentional
1501	1,370	2501	685	Utility issue fee (or reissue)
1502	490	2502	245	Design issue fee
1503	660	2503	330	Plant issue fee
1460	130	1460	130	Petitions to the Commissioner
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)
1806	180	1806	180	Submission of Information Disclosure Stmt
8021	40	8021	40	Recording each patent assignment per property (times number of properties)
1809	790	2809	395	Filing a submission after final rejection (37 CFR 1.129(a))
1810	790	2810	395	For each additional invention to be examined (37 CFR 1.129(b))
1801	790	2801	395	Request for Continued Examination (RCE)
1802	900	1802	900	Request for expedited examination of a design application

Other fee (specify) _____

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) **(\$ 0)**

CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]

I hereby certify that this correspondence is being:

- deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop _____, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450
- transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) _____.

Date _____

Signature _____

Typed or printed name _____

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee (\$)	Small Entity Fee Code	Small Entity Fee (\$)	Fee Description	Fee Paid
1001	790	2001	395	Utility filing fee	790.00
1002	350	2002	175	Design filing fee	
1003	550	2003	275	Plant filing fee	
1004	790	2004	395	Reissue filing fee	
1005	160	2005	80	Provisional filing fee	

SUBTOTAL (1) **\$790.00**

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

Total Claims	Extra Claims	Fee from below	Fee Paid
39	-20** = 19	X 18.00	= \$342.00
Independent Claims	3	-3** =	X = 0
Multiple Dependent		X	= 0

Large Entity Fee Code	Large Entity Fee (\$)	Small Entity Fee Code	Small Entity Fee (\$)	Fee Description
1202	18	2202	9	Claims in excess of 20
1201	88	2201	44	Independent claims in excess of 3
1203	300	2203	150	Multiple dependent claim, if not paid
1204	88	2204	44	** Reissue independent claims over original patent
1205	18	2205	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) **\$342.00**

**or number previously paid, if greater; For Reissues, see above

SUBMITTED BY

Name (Print/Type)	Carlos R. Villamar	Registration No. (Attorney/Agent)	43,224	Telephone	(202) 585-8204
Signature		Date	October 4, 2004		

Complete (if applicable)

SEND TO: Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
 United States Patent and Trademark Office
 Address: COMMISSIONER FOR PATENTS
 P.O. Box 1450
 Alexandria, Virginia 22313-1450
 www.uspto.gov

APPLICATION NUMBER	FILING OR 371 (c) DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
10/956,070	10/04/2004	Xin Wang	111325-235000

22204
 NIXON PEABODY, LLP
 401 9TH STREET, NW
 SUITE 900
 WASHINGTON, DC 20004-2128

CONFIRMATION NO. 8299

FORMALITIES LETTER



OC000000014862240

Date Mailed: 01/03/2005

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

*Filing Date Granted***Items Required To Avoid Abandonment:**

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given **TWO MONTHS** from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The signature of the following inventor(s) is missing from the oath or declaration:
Mai Nguyen, Eddie J. Chen
- To avoid abandonment, a late filing fee or oath or declaration surcharge as set forth in 37 CFR 1.16(e) of \$130 for a non-small entity, must be submitted with the missing items identified in this letter.


SUMMARY OF FEES DUE:

Total additional fee(s) required for this application is **\$130** for a Large Entity

- **\$130** Late oath or declaration Surcharge.

Replies should be mailed to: Mail Stop Missing Parts
 Commissioner for Patents
 P.O. Box 1450
 Alexandria VA 22313-1450

*A copy of this notice **MUST** be returned with the reply.*


Customer Service Center
Initial Patent Examination Division (703) 308-1202

PART 3 - OFFICE COPY



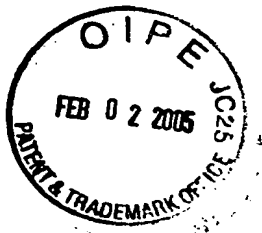
IPW

TRANSMITTAL FORM <i>(to be used for all correspondence after initial filing)</i>	Application Number	10/956,070
	Filing Date	October 4, 2004
	First Named Inventor	Mai NGUYEN, et al.
	Group Art Unit	3621
	Examiner Name	Not Yet Assigned
Total Number of Pages in This Submission	Attorney Docket Number	111325-235000

ENCLOSURES (check all that apply)		
<input type="checkbox"/> Fee Transmittal Form <input type="checkbox"/> Fee Attached <input type="checkbox"/> Amendment / Reply <input type="checkbox"/> After Final <input type="checkbox"/> Affidavits/declaration(s) <input type="checkbox"/> Extension of Time Request <input type="checkbox"/> Express Abandonment Request <input type="checkbox"/> Information Disclosure Statement <input type="checkbox"/> Certified Copy of Priority Document(s) <input type="checkbox"/> Response to Missing Parts/ Incomplete Application <input type="checkbox"/> Response to Missing Parts under 37 CFR 1.52 or 1.53	<input type="checkbox"/> Assignment Papers (for an Application) <input type="checkbox"/> Drawing(s) <input type="checkbox"/> Declaration and Power of Attorney <input type="checkbox"/> Licensing-related Papers <input type="checkbox"/> Petition <input type="checkbox"/> Petition to Convert to a Provisional Application <input type="checkbox"/> Power of Attorney, Revocation Change of Correspondence Address <input type="checkbox"/> Terminal Disclaimer <input type="checkbox"/> Request for Refund <input type="checkbox"/> CD, Number of CD(s) _____	<input type="checkbox"/> After Allowance Communication to Group <input type="checkbox"/> Appeal Communication to Board of Appeals and Interferences <input type="checkbox"/> Appeal Communication to Group (Appeal Notice, Brief, Reply Brief) <input type="checkbox"/> Proprietary Information <input type="checkbox"/> Status Letter <input checked="" type="checkbox"/> Application Data Sheet <input checked="" type="checkbox"/> Request for Corrected Filing Receipt with Enclosures <input type="checkbox"/> A self-addressed prepaid postcard for acknowledging receipt <input type="checkbox"/> Other Enclosure(s) (please identify below):
Remarks		<input checked="" type="checkbox"/> The Commissioner is hereby authorized to charge any additional fees required or credit any overpayments to Deposit Account No. 19-2380 for the above identified docket number.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT	
Firm or Individual name	Marc S. Kaufman Registration No. 35,212 Nixon Peabody LLP 401 9 th Street, N.W., Suite 900 Washington, D.C. 20004-2128
Signature	
Date	February 2, 2005

CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]	
I hereby certify that this correspondence is being:	
<input type="checkbox"/> deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop _____, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450	
<input type="checkbox"/> transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) _____.	
Date	Signature
_____	_____
	Typed or printed name



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of)	Confirmation No.: 8299
Mai NGUYEN, <i>et al.</i>)	Group Art Unit: 3621
Application No. 10/956,070)	Examiner: Unassigned
Filed: October 4, 2004)	
For: SYSTEM AND METHOD FOR)	
RIGHTS OFFERING AND GRANTING)	
USING SHARED STATE VARIABLES)	

U.S. Patent and Trademark Office
Customer Window, Mail Stop
Randolph Building
Alexandria, VA 22314

REQUEST FOR CORRECTED OFFICIAL FILING RECEIPT

Sir:

Please note that upon reviewing the Official Filing Receipt received in the above-identified application, the domestic priority data information was listed incorrectly and the order of the inventors were listed incorrectly. Please list the priority data as follows:

**--This application is a CIP of 10/162,212 06/05/2002
which is a CIP of 09/867,745 05/31/2001 PAT 6,754,642 --**

Please list the inventors in the following order:

**- -Mai Nguyen, Buena Park, CA
Xin Wang, Torrance, CA
Eddie J. Chen, Rancho Palos Verdes, CA
Bijan Tadayon, Germantown, MD --**

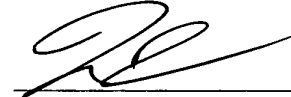
Submitted herewith is a copy of the original Official Filing Receipt indicating these corrections marked in red. Also, enclosed is a copy of application data sheet indicating that this error was due to an oversight at the USPTO.

Application No.: 10/956,070
Attorney Docket No.: 111325-235000

In view of the above, it is requested that a Corrected Filing Receipt be issued.

Respectfully submitted,

NIXON PEABODY, LLP



Marc S. Kaufman

Registration No. 35,212

Date: February 2, 2005

NIXON PEABODY LLP

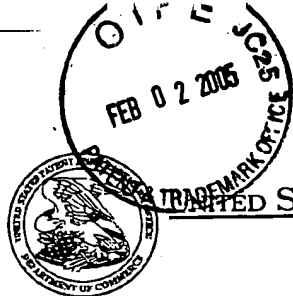
Customer No. 22204

401 Ninth Street, N.W., Suite 900

Washington, DC 20004

Telephone: (202) 585-8000

Fax:: (202) 585-8080



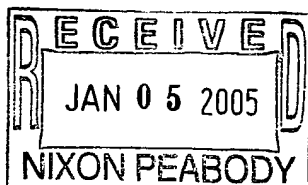
UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
 United States Patent and Trademark Office
 Address: COMMISSIONER FOR PATENTS
 P.O. Box 1450
 Alexandria, Virginia 22313-1450
 www.uspto.gov

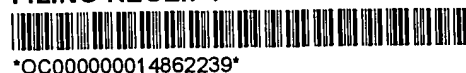
APPL NO.	FILING OR 371 (c) DATE	ART UNIT	FIL FEE REC'D	ATTY. DOCKET NO	DRAWINGS	TOT CLMS	IND CLMS
10/956,070	10/04/2004	3621	1132	111325-235000	17	39	3

CONFIRMATION NO. 8299

22204
 NIXON PEABODY, LLP
 401 9TH STREET, NW
 SUITE 900
 WASHINGTON, DC 20004-2128



FILING RECEIPT



OC000000014862239

Date Mailed: 01/03/2005

Receipt is acknowledged of this regular Patent Application. It will be considered in its order and you will be notified as to the results of the examination. Be sure to provide the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION when inquiring about this application. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. If an error is noted on this Filing Receipt, please write to the Office of Initial Patent Examination's Filing Receipt Corrections, facsimile number 703-746-9195. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections (if appropriate).

Applicant(s)

Xin Wang, Torrance, CA;
 Bijan Tadayon, Germantown, MD;
 Mai Nguyen, Buena Park, CA;
 Eddie J. Chen, Rancho Palos Verdes, CA;

Mai Nguyen
Xin Wang
Eddie J. Chen
Bijan Tadayon DOCKETED

By _____
 Nixon Peabody, LLP

Power of Attorney: None

Domestic Priority data as claimed by applicant

This application is a ~~cont~~ ^{CIP} of 10/162,212 06/05/2002
 which is a CIP of 09/867,745 05/31/2001 PAT 6,754,642

Foreign Applications

If Required, Foreign Filing License Granted: 01/03/2005

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US10/956,070**

Projected Publication Date: To Be Determined - pending completion of Missing Parts

Non-Publication Request: No

Best Available Copy

Early Publication Request: No

Title

System and method for rights offering and granting using shared state variables

Preliminary Class

705

**LICENSE FOR FOREIGN FILING UNDER
Title 35, United States Code, Section 184
Title 37, Code of Federal Regulations, 5.11 & 5.15**

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Office of Export Administration, Department of Commerce (15 CFR 370.10 (j)); the Office of Foreign Assets Control, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

Best Available Copy



APPLICATION DATA SHEET

Electronic Version 0.0.11

Stylesheet Version: 1.0

Attorney Docket Number: 111325-235000

Publication Filing Type: new-utility

Application Type: utility

Title of Invention: SYSTEM AND METHOD FOR RIGHTS OFFERING AND GRANTING USING SHARED STATE VARIABLES

Legal Representative:

Attorney or Agent: Marc S. Kaufman

Registration Number: 35212

Attorney or Agent: Carlos R. Villamar

Registration Number: 43224

Customer Number Correspondence Address: 22204

22204

Continuity Data:

This application is a continuation in part of 10/162,212 2002-06-05 which is a continuation-in-part application of application serial no. 09/867,745 filed on May 31, 2001 and which claims benefit from U.S. Provisional Application Serial No. 60/296,113 filed on June 7, 2001, U.S. Provisional Serial No. 60/331,625 filed on November 20, 2001 and U.S. Provisional Application Serial No. 60/331,624 filed on November 20, 2001

INVENTOR(s):

Primary Citizenship: United States
Given Name: Mai
Family Name: Nguyen
Residence City: Buena Park
Residence State: CA
Residence Country: US
Address: 5611 Cambridge Avenue
Buena Park CA, 96021 US

Primary Citizenship: United States
Given Name: Xin
Family Name: Wang
Residence City: Torrance
Residence State: CA
Residence Country: US
Address: 3720 Emerald Street, #V2
Torrance CA, US

Primary Citizenship: United States
Given Name: Eddie
Middle Name: J.
Family Name: Chen
Residence City: Rancho Palos Verdes
Residence State: CA
Residence Country: US
Address: 6796 Vallon Drive
Rancho Palos Verdes CA, 90275 US

Primary Citizenship: United States
Given Name: Bijan
Family Name: Tadayon
Residence City: Germantown
Residence State: MD
Residence Country: US
Address: 20920 Scottsbury Drive
Germantown Maryland, 20876 US



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
 United States Patent and Trademark Office
 Address: COMMISSIONER FOR PATENTS
 P.O. Box 1450
 Alexandria, Virginia 22313-1450
 www.uspto.gov

APPLICATION NUMBER	FILING OR 371 (c) DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
10/956,070	10/04/2004	Xin Wang	111325-235000

CONFIRMATION NO. 8299

FORMALITIES LETTER



OC000000014862240

22204
 NIXON PEABODY, LLP
 401 9TH STREET, NW
 SUITE 900
 WASHINGTON, DC 20004-2128

Date Mailed: 01/03/2005

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

*Filing Date Granted***Items Required To Avoid Abandonment:**

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given **TWO MONTHS** from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The signature of the following inventor(s) is missing from the oath or declaration:
Mai Nguyen, Eddie J. Chen
- To avoid abandonment, a late filing fee or oath or declaration surcharge as set forth in 37 CFR 1.16(e) of \$130 for a non-small entity, must be submitted with the missing items identified in this letter.

SUMMARY OF FEES DUE:

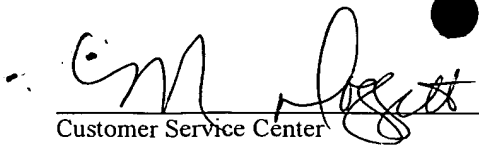
Total additional fee(s) required for this application is \$130 for a Large Entity

- \$130 Late oath or declaration Surcharge.

Replies should be mailed to: Mail Stop Missing Parts
 Commissioner for Patents
 P.O. Box 1450
 Alexandria VA 22313-1450

03/01/2005 SZENDIE1 00000071 192380 10956070
 01 FC:1051 130.00 DA

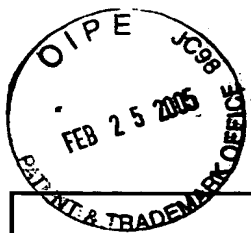
*A copy of this notice **MUST** be returned with the reply.*



Customer Service Center

Initial Patent Examination Division (703) 308-1202

PART 2 - COPY TO BE RETURNED WITH RESPONSE

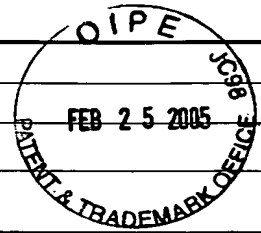


TRANSMITTAL FORM <i>(to be used for all correspondence after initial filing)</i>		Application Number	10/956,070
		Filing Date	October 4, 2004
		First Named Inventor	Mai NGUYEN et al.
		Group Art Unit	3621
		Confirmation No.	8299
Total Number of Pages in This Submission		Attorney Docket Number	111325-235000

ENCLOSURES <i>(check all that apply)</i>		
<input checked="" type="checkbox"/> Fee Transmittal Form <input type="checkbox"/> Fee Attached <input type="checkbox"/> Amendment / Reply <input type="checkbox"/> After Final <input type="checkbox"/> Affidavits/declaration(s) <input type="checkbox"/> Extension of Time Request <input type="checkbox"/> Express Abandonment Request <input type="checkbox"/> Information Disclosure Statement <input type="checkbox"/> Certified Copy of Priority Document(s) <input checked="" type="checkbox"/> Notice to File Missing Parts of Nonprovisional Application Formalities Letter <input type="checkbox"/> Response to Missing Parts under 37 CFR 1.52 or 1.53	<input checked="" type="checkbox"/> Assignment Papers <i>(for an Application)</i> <input type="checkbox"/> Drawing(s) <input checked="" type="checkbox"/> Declaration and Power of Attorney <input type="checkbox"/> Licensing-related Papers <input type="checkbox"/> Petition <input type="checkbox"/> Petition to Convert to a Provisional Application <input type="checkbox"/> Power of Attorney, Revocation Change of Correspondence Address <input type="checkbox"/> Terminal Disclaimer <input type="checkbox"/> Request for Refund <input type="checkbox"/> CD, Number of CD(s) _____	<input type="checkbox"/> After Allowance Communication to Group <input type="checkbox"/> Appeal Communication to Board of Appeals and Interferences <input type="checkbox"/> Appeal Communication to Group <i>(Appeal Notice, Brief, Reply Brief)</i> <input type="checkbox"/> Proprietary Information <input type="checkbox"/> Status Letter <input type="checkbox"/> Application Data Sheet <input type="checkbox"/> Request for Corrected Filing Receipt with Enclosures <input type="checkbox"/> A self-addressed prepaid postcard for acknowledging receipt <input type="checkbox"/> Other Enclosure(s) <i>(please identify below):</i>
Remarks	<input checked="" type="checkbox"/> The Commissioner is hereby authorized to charge any additional fees required or credit any overpayments to Deposit Account No. 19-2380 for the above identified docket number.	

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT	
Firm or Individual name	Marc S. Kaufman Nixon Peabody LLP 401 9 th Street, N.W. Suite 900 Washington, D.C. 20004-2128
Signature	
Date	February 24, 2005

CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]	
I hereby certify that this correspondence is being:	
<input type="checkbox"/> deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop _____, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450	
<input type="checkbox"/> transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) _____	
Date	Signature
_____	_____
	Typed or printed name



Effective on 12/08/2004.
Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).

FEE TRANSMITTAL FOR FY 2005

<i>Complete if Known</i>	
Application Number	10/956,070
Filing Date	October 4, 2004
First Named Inventor	Mai NGUYEN et al.
<input type="checkbox"/> Applicant claims small entity status. See 37 CFR 1.27	Confirmation No. 8299
TOTAL AMOUNT OF PAYMENT	\$170.00
	Art Unit 3621
	Attorney Docket No. 111325-235000

METHOD OF PAYMENT (check all that apply)

Check Credit Card Money Order None Other (please identify): _____

Deposit Account Deposit Account Number: 19-2380 (111325-235000) Deposit Account Name: Nixon Peabody LLP

For the above-identified deposit account, the Director is hereby authorized to: (check all that apply)

Charge fee(s) indicated below Charge fee(s) indicated below, except for the filing fee

Charge any additional fee(s) or underpayments of fee(s) under 37 CFR 1.16 and 1.17 Credit any overpayments

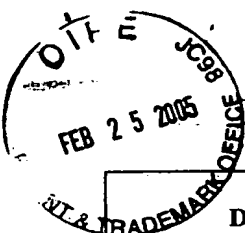
WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-20238.

FEE CALCULATION

1. BASIC FILING, SEARCH AND EXAMINATION FEES							
	FILING FEES		SEARCH FEES		EXAMINATION FEES		
		<u>Small Entity</u>		<u>Small Entity</u>		<u>Small Entity</u>	
<u>Application Type</u>	<u>Fee (\$)</u>	<u>Fee (\$)</u>	<u>Fee (\$)</u>	<u>Fee (\$)</u>	<u>Fee (\$)</u>	<u>Fee (\$)</u>	<u>Fees Paid (\$)</u>
Utility	300	150	500	250	200	100	\$
Design	200	100	100	50	130	65	\$
Plant	200	100	300	150	160	80	\$
Reissue	300	150	500	250	600	300	\$
Provisional	200	100	0	0	0	0	\$
2. EXCESS CLAIM FEES							
<u>Fee Description</u>						<u>Fee (\$)</u>	<u>Small Entity Fee (\$)</u>
Each claim over 20 or, for Reissues, each claim over 20 and more than in the original patent						50	25
Each independent claim over 3 or, for Reissues, each independent claim more than in the original patent						200	100
Multiple document claims						360	180
<u>Total Claims</u>	<u>Extra Claims</u>		<u>Fee (\$)</u>	<u>Fee Paid (\$)</u>	<u>Multiple Dependent Claims</u>	<u>Fee (\$)</u>	<u>Fee Paid (\$)</u>
- 20 or HP =	x		=				
HP -- highest number of total claims paid for, if greater than 20							
<u>Indep. Claims</u>	<u>Extra Claims</u>		<u>Fee (\$)</u>	<u>Fee Paid (\$)</u>			
- 3 or HP =	x		=				
HP -- highest number of independent claims paid for, if greater than 3							
3. APPLICATION SIZE FEE							
If the specification and drawings exceed 100 sheets of paper, the application size fee due is \$250 (\$125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).							
<u>Total Sheets</u>	<u>Extra Sheets</u>		<u>Number of each additional 50 or fraction thereof</u>	<u>Fee (\$)</u>	<u>Fee Paid (\$)</u>		
60	- 100 =	0	/ 50 =	(round up to a whole number)	x		=
4. OTHER FEE(S)							
							<u>Fees Paid (\$)</u>
Non-English Specification,	\$130 fee (no small entity discount)						\$130.00
Assignment Recordation Fee	\$40 fee (no small entity discount)						\$40.00

SUBMITTED BY

Signature	Marc S. Kaufman	Registration No. (Attorney/Agent)	35,212	Telephone	(202) 585-8000
Name (Print/Type)				Date	February 24, 2005



DECLARATION (37 CFR 1.63) FOR UTILITY OR DESIGN APPLICATION USING AN APPLICATION DATA SHEET (37 CFR 1.76) AND POWER OF ATTORNEY

Docket No. 111325-235000

Title of Invention: SYSTEM AND METHOD FOR RIGHTS OFFERING AND GRANTING USING SHARED STATE VARIABLES

As the below named inventor(s), I/we declare that:

This declaration is directed to:

- The attached application, or
[X] Application No. 10/956,070, filed on October 4, 2004,
as amended on (if applicable);

I/We believe that I/we am/are the original and first inventor(s) of the subject matter which is claimed and for which a patent is sought;

I/We have reviewed and understand the contents of the above-identified application, including the claims, as amended by any amendment specifically referred to above;

I/We acknowledge the duty to disclose to the United States Patent and Trademark Office all information known to me/us to be material to patentability as defined in 37 CFR 1.56, including for continuation-in-part applications, material information which became available between the filing date of the prior application and the national or PCT International filing date of the continuation-in-part application.

All statements made herein of my/own knowledge are true, all statements made herein on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under 18 U.S.C. 1001, and may jeopardize the validity of the application or any patent issuing thereon.

I/We hereby appoint:

Practitioners at Customer Number 22204 as my/our attorney(s) or agent(s) to prosecute the application identified above, and to transact all business in the United States Patent and Trademark Office connected therewith.

FULL NAME OF INVENTOR(S)

Inventor one: Mai NGUYEN

Signature: [Handwritten Signature] (2/7/2005) Citizen of: UNITED STATES

Inventor two: Xin WANG

Signature: [Handwritten Signature] (2-7-05) Citizen of: UNITED STATES

Inventor three: Eddie J. CHEN

Signature: [Handwritten Signature] 2/7/2005 Citizen of: UNITED STATES

Inventor four: Bijan TADAYON

Signature: [Handwritten Signature] (Feb-2-2005) Citizen of: UNITED STATES

Additional inventors are being named on additional form(s) attached hereto.



ZPW

TRANSMITTAL FORM <i>(to be used for all correspondence after initial filing)</i>	Application Number	10/956,070
	Filing Date	October 4, 2004
	First Named Inventor	Mai NGUYEN, et al.
	Group Art Unit	3621
	Examiner Name	Not Yet Assigned
Total Number of Pages in This Submission	Attorney Docket Number	111325-235000

ENCLOSURES (check all that apply)		
<input type="checkbox"/> Fee Transmittal Form <input type="checkbox"/> Fee Attached <input type="checkbox"/> Amendment / Reply <input type="checkbox"/> After Final <input type="checkbox"/> Affidavits/declaration(s) <input type="checkbox"/> Extension of Time Request <input type="checkbox"/> Express Abandonment Request <input checked="" type="checkbox"/> Information Disclosure Statement <input type="checkbox"/> Certified Copy of Priority Document(s) <input type="checkbox"/> Response to Missing Parts/ Incomplete Application <input type="checkbox"/> Response to Missing Parts under 37 CFR 1.52 or 1.53	<input type="checkbox"/> Assignment Papers (for an Application) <input type="checkbox"/> Drawing(s) <input type="checkbox"/> Declaration and Power of Attorney <input type="checkbox"/> Licensing-related Papers <input type="checkbox"/> Petition <input type="checkbox"/> Petition to Convert to a Provisional Application <input type="checkbox"/> Power of Attorney, Revocation Change of Correspondence Address <input type="checkbox"/> Terminal Disclaimer <input type="checkbox"/> Request for Refund <input type="checkbox"/> CD, Number of CD(s) _____	<input type="checkbox"/> After Allowance Communication to Group <input type="checkbox"/> Appeal Communication to Board of Appeals and Interferences <input type="checkbox"/> Appeal Communication to Group (Appeal Notice, Brief, Reply Brief) <input type="checkbox"/> Proprietary Information <input type="checkbox"/> Status Letter <input type="checkbox"/> Application Data Sheet <input type="checkbox"/> Request for Corrected Filing Receipt with Enclosures <input type="checkbox"/> A self-addressed prepaid postcard for acknowledging receipt <input type="checkbox"/> Other Enclosure(s) (please identify below):
Remarks	<input checked="" type="checkbox"/> The Commissioner is hereby authorized to charge any additional fees required or credit any overpayments to Deposit Account No. 19-2380 for the above identified docket number.	

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT	
Firm or Individual name	Carlos R. Villamar Registration No. 43,224 Nixon Peabody LLP 401 9 th Street, N.W., Suite 900 Washington, D.C. 20004-2128
Signature	
Date	8/2/05

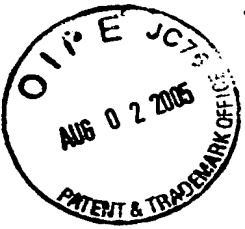
CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]

I hereby certify that this correspondence is being:

deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop _____, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450

transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) _____.

_____ Date _____ Signature _____
 _____ Typed or printed name _____



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:

Mai NGUYEN, *et al.*)

Serial No. 10/956,070)

Filed: October 4, 2004)

For: **SYSTEM AND METHOD FOR RIGHTS**)

OFFERING AND GRANTING USING)

SHARED STATE VARIABLES)

Examiner: Net Yet Assigned
Group Art: 3621

U.S. Patent and Trademark Office
Customer Service Window
Randolph Building
401 Dulany Street
Alexandria, VA 22314

INFORMATION DISCLOSURE STATEMENT

Sir:

In accordance with the duty of disclosure as set forth in 37 C.F.R. §1.56, Applicants hereby submit the following information in conformance with 37 C.F.R. §§ 1.97 and 1.98. The references listed on the attached PTO-1449 forms have been made of record in parent application serial number 10/162,212, Filed on June 5, 2002, therefore no copies of the references cited are submitted herewith.

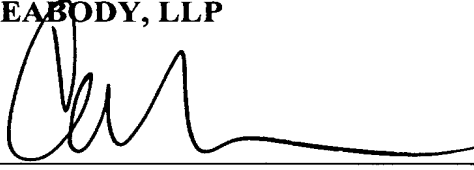
It is requested that the accompanying PTO-1449 be considered and made of record in the above-identified application. To assist the Examiner, the documents are listed on the attached form PTO-1449. It is respectfully requested that an Examiner initial a copy of this form be returned to the undersigned.

The Commissioner is hereby authorized to charge any fees connected with this filing which may be required now, or credit any overpayment to Deposit Account No. 19-2380 (111325-235000).

Respectfully submitted,

NIXON PEABODY, LLP

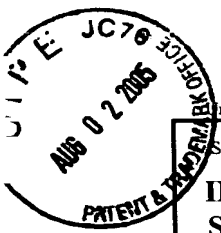
By: _____



Carlos R. Villamar
Registration No. 43,224

Date: 3/2/05

NIXON PEABODY LLP
401 9th Street, N.W., Suite 900
Washington, DC 20004-2128
Telephone: (202) 585-8000



Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

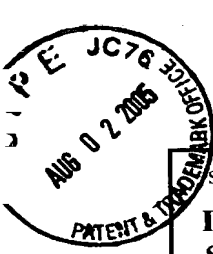
Substitute for form 1449A/PTO		<i>Complete if Known</i>	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>		Application Number	10/956,070
		Filing Date	October 4, 2004
		First Named Inventor	Mai NGUYEN, <i>et al.</i>
		Art Unit	3621
		Examiner Name	Not Yet Assigned
Sheet	1	of	10
		Attorney Docket Number	111325-235000

U.S. PATENT DOCUMENTS						
Examiner Initials*	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ² (if known)				
		US-3,263,158		07/01/1996	Janis	
		US-3,609,697		09/28/1971	Blevins et al.	
		US-3,790,700		02/05/1974	Callais et al.	
		US-3,798,605		03/19/1974	Feistel	
		US-4,159,468		06/26/1979	Barnes et al.	
		US-4,220,991		09/02/1980	Hamano et al.	
		US-4,278,837		07/14/1981	Best	
		US-4,323,921		04/06/1982	Guillou	
		US-4,442,486		04/10/1984	Mayer	
		US-4,529,870		07/16/1985	Chaum	
		US-4,558,176		12/10/1985	Arnold et al.	
		US-4,593,376		06/03/1986	Volk	
		US-4,614,861		09/30/1986	Pavlov et al.	
		US-4,644,493		02/17/1987	Chandra et al.	
		US-4,658,093		04/14/1987	Hellman	
		US-4,713,753		12/15/1987	Beobert et al.	
		US-4,740,890		04/26/1988	William	
		US-4,796,220		01/03/1989	Wolfe	
		US-4,817,140		03/28/1989	Chandra et al.	
		US-4,827,508		05/02/1989	Shear	
		US-4,868,376		09/19/1989	Lessin et al.	

FOREIGN PATENT DOCUMENTS							
Examiner Initials*	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
		Country Code ³ Number ⁴	Kind Code ⁵ (if known)				
		0 332 304 A3	EP	09/13/1989			
		0 084 441	EP	07/27/1983	TABS LIMITED		
		0 180 460	EP	05/07/1986	SONY CORPORATION		
		0 332 707	EP	09/01/1989	HONDA GIKEN KOGYO KABUSHIKI KAISHA		
		0 651 554	EP	05/03/1995	EASTMAN KODAK CO.		
		0 668 695	EP	08/23/1995	VICTOR COMPANY OF JAPAN LIMITED		
		0 715 244 A	EP	06/05/1996			
		0 715 243 A	EP	06/05/1996			
		0 725 376	EP	08/07/1996	SONY CORP.		
		0 731 404 A1	EP	09/09/1996			
		0 818 748 A2		01/14/1998			

Examiner Signature	Date Considered
--------------------	-----------------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.
¹ Applicant's unique citation designation number (optional). ² See Kind Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.
 Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.



Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO				Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Application Number	10/956,070
				Filing Date	October 4, 2004
				First Named Inventor	Mai NGUYEN, <i>et al.</i>
				Art Unit	3621
				Examiner Name	Not Yet Assigned
Sheet	2	of	10	Attorney Docket Number	111325-235000

U.S. PATENT DOCUMENTS						
Examiner Initials*	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ² (if known)				
		US-4,891,838		01/02/1990	Faber	
		US-4,924,378		05/08/1990	Hershey et al.	
		US-4,932,054		06/05/1990	Chou et al.	
		US-4,937,863		06/26/1990	Robert et al.	
		US-4,949,187		08/14/1990	Cohen	
		US-4,953,209		08/28/1990	Ryder, Sr. et al.	
		US-4,961,142		10/02/1990	Elliott et al.	
		US-4,975,647		12/04/1990	Downer et al.	
		US-4,977,594		12/11/1990	Shear	
		US-4,999,806		03/12/1991	Chernow et al.	
		US-5,010,571		04/23/1991	Katznelson	
		US-5,014,234		05/07/1991	Edwards, Jr.	
		US-5,023,907		06/11/1991	Johnson et al.	
		US-5,047,928		09/10/1991	Wiedemer	
		US-5,050,213		09/17/1991	Shear	
		US-5,052,040		09/24/1991	Preston et al.	
		US-5,058,164		10/15/1991	Elmer et al.	
		US-5,103,476		04/07/1992	Waite et al.	
		US-5,113,519		05/12/1992	Johnson et al.	
		US-5,136,643		08/04/1992	Fischer	

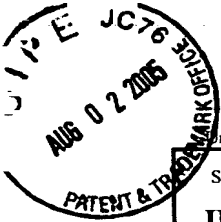
FOREIGN PATENT DOCUMENTS							
Examiner Initials*	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
		Country Code ³	Number ⁴ Kind Code ⁵ (if known)				
		05-268415	JP	10/15/1993	RICOH CO LTD		Abst
		06-175794	JP	06/24/1994	FUJI XEROX CO LTD		Abst
		06-215010	JP	08/05/1994	SONY CORP.		Abst
		07-084852	JP	03/31/1995	HITACHI LTD.		Abst
		07-200317	JP	08/04/1995	TOSHIBA CORP.		Abst
		07-244639	JP	09/19/1995	FUJITSU LTD		Abst
		62-241061	JP	10/21/1987	NEC CORP.		Abst
		64-068835	JP	03/14/1989	RYOICHI MORI		Abst
		WO 00/08909	A	02/24/2000			
		WO 01 13198	A	01/22/2001			
		WO 01/63528	PCT	08/30/2001	IPDN COPR.		
		WO 92/20022	PCT	11/12/1992	DIGITAL EQUIPMENT CORP.		

Examiner Signature	Date Considered
--------------------	-----------------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.



Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO			Complete if Known		
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>			Application Number	10/956,070	
			Filing Date	October 4, 2004	
			First Named Inventor	Mai NGUYEN, <i>et al.</i>	
			Art Unit	3621	
			Examiner Name	Not Yet Assigned	
Sheet	3	of	10	Attorney Docket Number	111325-235000

U.S. PATENT DOCUMENTS						
Examiner Initials ¹	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ² (if known)				
		US-5,138,712		08/11/1992	Corbin	
		US-5,146,499		09/08/1992	Geffrotin	
		US-5,148,481		09/15/1992	Abraham et al.	
		US-5,159,182		10/27/1992	Eisele	
		US-5,183,404		02/02/1993	Aldous et al.	
		US-5,191,193		03/02/1993	Le Roux	
		US-5,204,897		04/20/1993	Wyman	
		US-5,222,134		06/22/1993	Waite et al.	
		US-5,235,642		08/10/1993	Wobber et al.	
		US-5,247,575		09/21/1993	Sprague et al.	
		US-5,255,106		10/19/1993	Castro	
		US-5,260,999		11/09/1993	Wyman	
		US-5,263,157		11/16/1993	Janis	
		US-5,263,158		11/16/1993	Janis	
		US-5,276,444		01/04/1994	McNair	
		US-5,276,735		01/04/1994	Boebert et al.	
		US-5,291,596		03/01/1994	Mita	
		US-5,301,231		04/05/1994	Abraham et al.	
		US-5,311,591		05/10/1994	Fischer	
		US-5,319,705		06/07/1994	Halter et al.	

FOREIGN PATENT DOCUMENTS							
Examiner Initials ¹	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
		Country Code ³ Number ⁴	Kind Code ⁵ (if known)				
		WO 93/01550	PCT	01/21/1993	INFOLOGIC SOFTWARE, INC		
		WO 94/01821	PCT	01/20/1994	SECURE COMPUTING CORP.		
		WO 96/24092	PCT	08/08/1996	BENSON, Greg		
		WO 97/48203	PCT	12/18/1997	INTEL CORP.		
		WO 98/11690	PCT	03/19/1998	GLOVER, John J.		
		WO 98/42098	PCT	09/24/1998	CRYPTOWORKS, INC.		
		WO 99/49615	PCT	09/30/1999	MICROTOME		
		WO 00/73922	A2 PCT	12/07/2000			
		WO 01/24530	A2 PCT	04/05/2001			
		WO 00/59152	PCT	10/05/2000			
Examiner Signature					Date Considered		

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO				Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Application Number	10/956,070
				Filing Date	October 4, 2004
				First Named Inventor	Mai NGUYEN, <i>et al.</i>
				Art Unit	3621
				Examiner Name	Not Yet Assigned
Sheet	4	of	10	Attorney Docket Number	111325-235000

U.S. PATENT DOCUMENTS						
Examiner Initials [*]	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ² (if known)				
		US-5,337,357		08/09/1994	Chou et al.	
		US-5,386,369		01/31/1995	Christiano	
		US-5,339,091		08/16/1994	Yamazaki et al.	
		US-5,341,429		08/23/1994	Stringer et al.	
		US-5,347,579		09/13/1994	Blandford	
		US-5,381,526		01/10/1995	Ellson	
		US-5,394,469		02/28/1995	Nagel et al.	
		US-5,410,598		04/25/1995	Shear	
		US-5,412,717		05/02/1995	Fischer	
		US-5,428,606		06/27/1995	Moskowitz	
		US-5,432,849		07/11/1995	Johnson et al.	
		US-5,438,508		08/01/1995	Wyman	
		US-5,444,779		08/22/1995	Daniele	
		US-5,453,601		09/26/1995	Rosen	
		US-5,455,953		10/03/1995	Russell	
		US-5,457,746		10/10/1995	Dolphin	
		US-5,473,687		12/05/1995	Lipscomb et al.	
		US-5,473,692		12/05/1995	Davis	
		US-5,499,298		03/12/1996	Narasimhalu et al.	
		US-5,502,766		03/26/1996	Boebert et al.	
		US-5,504,814		04/02/1996	Miyahara	

FOREIGN PATENT DOCUMENTS							
Examiner Initials [*]	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
		Country Code ³ Number ⁴	Kind Code ² (if known)				
		0 715 246 A	EP	06/05/1996			
		1 041 823 A2	EP	10/04/2000			
		2 136 175	GB	09/12/1984	ATALLA CORP.		
		2 236 604	GB	04/10/1991	SUN MICROSYSTEMS INC		
		0 715 241	JP	06/05/1996	MITSUBISHI CORP.		
		04-369068	JP	12/21/1992	CHIYUUBU NIHON DENKI SOFUTOUEA KK		Abst

Examiner Signature	Date Considered
--------------------	-----------------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				<i>Complete if Known</i>		
				Application Number	10/956,070	
Sheet		5	of	10	Filing Date	October 4, 2004
					First Named Inventor	Mai NGUYEN, et al.
					Art Unit	3621
					Examiner Name	Not Yet Assigned
					Attorney Docket Number	111325-235000

U.S. PATENT DOCUMENTS						
Examiner Initials ⁷	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ² (if known)				
		US-5,504,818		04/02/1996	Okano	
		US-5,504,837		04/02/1996	Griffeth et al.	
		US-5,509,070		04/16/1996	Schull	
		US-5,530,235		06/25/1996	Stefik et al.	
		US-5,532,920		07/02/1996	Hartrick et al.	
		US-5,534,975		07/09/1996	Stefik et al.	
		US-5,539,735		07/23/1996	Moskowitz	
		US-5,563,946		10/08/1996	Cooper et al.	
		US-5,568,552		10/22/1996	Davis	
		US-5,621,797		04/15/1997	Rosen	
		US-5,629,980		05/13/1997	Stefik et al.	
		US-5,633,932		05/27/1997	Davis et al.	
		US-5,634,012		05/27/1997	Stefik et al.	
		US-5,638,443		06/10/1997	Stefik et al.	
		US-5,649,013		07/15/1997	Stuckey et al.	
		US-5,655,077		08/05/1997	Jones et al.	
		US-5,671,412		09/1997	Christiano	
		US-5,708,717		01/13/1998	Alasia	
		US-5,734,823		03/31/1998	Saigh et al.	
		US-5,734,891		03/31/1998	Saigh	
		US-5,737,413		04/07/1998	Akiyama et al.	

FOREIGN PATENT DOCUMENTS								
Examiner Initials ⁷	Cite No. ¹	Foreign Patent Document			Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
		Country Code ³	Number ⁴	Kind Code ⁵ (if known)				

Examiner Signature	Date Considered
--------------------	-----------------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Complete if Known		
				Application Number	10/956,070	
Sheet		6	of	10	Filing Date	October 4, 2004
					First Named Inventor	Mai NGUYEN, et al.
					Art Unit	3621
					Examiner Name	Not Yet Assigned
					Attorney Docket Number	111325-235000

U.S. PATENT DOCUMENTS						
Examiner Initials*	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ² (if known)				
		US-5,737,416		04/07/1998	Cooper et al.	
		US-5,745,569		04/28/1998	Moskowitz et al.	
		US-5,748,783		05/05/1998	Rhoads	
		US-5,757,907		05/26/1998	Cooper et al.	
		US-5,758,069		05/26/1998	Olsen	
		US-5,761,686		06/02/1998	Bloomberg	
		US-5,764,807		06/9/1998	Pearlman et al.	
		US-5,765,152		06/09/1998	Erickson	
		US-5,768,426		06/16/1998	Rhoads	
		US-5,790,664		08/1998	Coley et al.	
		US-5,794,207		08/11/1998	Walker et al.	
		US-5,825,892		10/20/1998	Braudaway et al.	
		US-5,848,154		12/08/1998	Nishio et al.	
		US-5,892,900		04/06/1999	Ginter et al.	
		US-5,910,987		06/08/1999	Ginter et al.	
		US-5,915,019		06/22/1999	Ginter et al.	
		US-5,917,912		06/29/1999	Ginter et al.	
		US-5,920,861		07/06/1999	Hall et al.	
		US-5,925,127		07/1999	Ahmad	
		US-5,940,504		08/17/1999	Griswold	
		US-5,943,422		08/24/1999	Van Wie et al.	
		US-5,949,876		09/07/1999	Ginter et al.	
		US-5,982,891		11/09/1999	Ginter et al.	
		US-5,991,306		11/23/1999	Burns, et al.	
		US-5,999,949		12/07/1999	Crandall	
		US-6,009,401		12/1999	Horstmann	
		US-6,047,067		04/04/2000	Rosen	
		US-6,056,786		05/2000	Rivera et al.	
		US-6,112,181		08/29/2000	Shear et al.	
		US-6,112,239		08/29/2000	Kenner et al.	

FOREIGN PATENT DOCUMENTS							
Examiner Initials*	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
		Country Code ³ Number ⁴	Kind Code ⁵ (if known)				

Examiner Signature		Date Considered	
--------------------	--	-----------------	--

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO				Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Application Number	10/956,070
				Filing Date	October 4, 2004
				First Named Inventor	Mai NGUYEN, <i>et al.</i>
				Art Unit	3621
				Examiner Name	Not Yet Assigned
Sheet	8	of	10	Attorney Docket Number	111325-235000

OTHER PRIOR ART - NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
		"National Semiconductor and EPR Partner for Information Metering/Data Security Cards" March 4, 1994, Press Release from Electronic Publishing Resources, Inc.	
		Weber, R., "Digital Rights Management Technology" October 1995	
		Flasche, U. et al., "Decentralized Processing of Documents", pp. 119-131, 1986, Comput. & Graphics, Vol. 10, No. 2	
		Mori, R. et al., "Superdistribution: The Concept and the Architecture", pp. 1133-1146, 1990. The Transactions of the IEICE, Vo. E 73, No. 7, Tokyo, JP	
		Weber, R., "Metering Technologies for Digital Intellectual Property", pp. 1-29, Oct. 1994, A Report to the International Federation of Reproduction Rights Organizations	
		Clark, P.C. et al., "Bits: A Smartcard protected Operating System", pp. 66-70 and 94, November 1994, Communications of the ACM, Vol. 37, No. 11	
		Ross, P.E., "Data Guard", pp. 101, June 6, 1994, Forbes	
		Saigh, W.K., "Knowledge is Sacred", 1992, Video Pocket/Page Reader Systems, Ltd.	
		Kahn, R.E., "Deposit, Registration and Recordation in an Electronic Copyright Management System", pp. 1-19, August 1992, Corporation for National Research Initiatives, Virginia	
		Hilts, P. et al., "Books While U Wait", pp. 48-50, January 3, 1994, Publishers Weekly	
		Strattner, A, "Cash Register on a Chip may Revolutionaize Software Pricing and Distribution; Wave Systems Corp.", pp. 1-3, April 1994, Computer Shopper, Vol. 14, No. 4, ISSN 0886-0556	

Examiner Signature	Date Considered	
--------------------	-----------------	--

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO				Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Application Number	10/956,070
				Filing Date	October 4, 2004
				First Named Inventor	Mai NGUYEN, <i>et al.</i>
				Art Unit	3621
				Examiner Name	Not Yet Assigned
				Attorney Docket Number	111325-235000
Sheet	9	of	10		

OTHER PRIOR ART - NON PATENT LITERATURE DOCUMENTS			
Examiner Initials [*]	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
		O'Conner, M., "New Distribution Option for Electronic Publishers; iOpener Data Encryption and Metering System for CD-ROM use; Column", pp. 1-6, March 1994, CD-ROM Professional, Vol. 7, No. 2, ISSN: 1409-0833	
		Willett, S., "Metered PCs: Is Your System Watching You? Wave System beta tests new technology", pp. 84, May 2, 1994, InfoWorld	
		Linn, R., "Copyright and Information Services in the Context of the National Research and Education Network", pp. 9-20, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Perrit, Jr., H., "Permission Headers and Contract Law", pp. 27-48, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Upthegrove, L., "Intellectual Property Header Descriptors: A Dynamic Approach", pp. 63-66, January 1994, IMA Intellectual Property Proceedings, Vol. 1, Issue 1	
		Sirbu, M., "Internet Billing Service Design and prototype Implementation", pp. 67-80, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Simmell, S. et al., "Metering and Licensing of Resources: Kala's General Purpose Approach", pp. 81-110, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Kahn, R., "Deposit, Registration and Recordation in an Electronic Copyright Management System", pp. 111-120, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Tygar, J. et al., "Dyad: A System for Using Physically Secure Coprocessors", pp. 121-152, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Griswold, G., "A Method for Protecting Copyright on Networks", pp. 169-178, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	

Examiner Signature		Date Considered	
-----------------------	--	--------------------	--

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO				Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Application Number	10/956,070
				Filing Date	October 4, 2004
				First Named Inventor	Mai NGUYEN, <i>et al.</i>
				Art Unit	3621
				Examiner Name	Not Yet Assigned
Sheet	10	of	10	Attorney Docket Number	111325-235000

OTHER PRIOR ART - NON PATENT LITERATURE DOCUMENTS			
Examiner Initials [*]	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
		Nelson, T., "A Publishing and Royalty Model for Networked Documents", pp. 257-259, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Robinson, E., "Redefining Mobile Computing", pp. 238-240, 247-248 and 252, July 1993, PC Computing	
		Abadi, M. et al., "Authentication and Delegation with Smart-cards", PP. 1-24, 1990, Research Report DEC Systems Research Center	
		Mark Stefik, "Letting Loose the Light: Igniting Commerce in Electronic Publication", pp. 219-253, 1996, Internet Dreams: Archetypes, Myths, and Metaphors, IDSN 0-262-19373-6	
		Mark Stefik, "Letting Loose the Light: Igniting Commerce in Electronic Publication", pp. 2-35, February 8, 1995, Internet Dreams: Archetypes, Myths and Metaphors	
		Henry H. Perritt, Jr., "Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment", April 2-3, 1993, Knowbots, Permissions Headers & Contract Law	

Examiner Signature		Date Considered	
-----------------------	--	--------------------	--

^{*} EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. **DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.**

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	1	("5715403").PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	OFF	2005/10/15 12:20
S2	1	("5715403").PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	OFF	2005/10/15 12:24
S3	0	S2 and (usage near right\$1) and (meta near2 right\$1) and licens\$ and state and variable	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	ON	2005/10/15 12:25
S4	0	S2 and (usage with right\$1) and (meta with right\$1) and licens\$ and state\$1 and variable\$1	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	ON	2005/10/15 12:23
S5	0	S2 and (usage with right\$1) and (meta) and licens\$ and state\$1 and variable\$1	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	ON	2005/10/15 12:23
S6	0	S2 and (usage with right\$1) and (meta) and licens\$	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	ON	2005/10/15 12:24
S7	0	S2 and right\$1 and (meta) and licens\$	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	ON	2005/10/15 12:24
S8	1	S2 and (usage near right\$1)	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	ON	2005/10/15 12:25
S9	0	S2 and (usage near right\$1) and meta	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	ON	2005/10/15 12:25
S10	1	S2 and (usage near right\$1) and licens\$3	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	ON	2005/10/15 12:25

S11	1	S2 and (usage near right\$1) and licens\$3 and state\$1 and variable\$1	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	ON	2005/10/15 12:26
S12	1	("6233684").PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	OFF	2005/10/15 12:29
S13	1	("5629980").PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	OFF	2005/10/15 12:30
S14	15	((Digital near Right\$1) near management) and (usage near right\$1) and (Meta near2 right\$1) and licens\$ and (state near1 variable\$1)	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	ON	2005/10/15 12:49
S15	1	("20030009423").PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	OFF	2005/10/15 12:49
S16	1	("6226618").PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; IBM_TDB	OR	OFF	2005/10/15 12:49
S17	61	("6226618").URPN.	USPAT	OR	ON	2005/10/15 14:38
S18	10	S17 and (usage near right\$1) and licens\$3	USPAT	OR	ON	2005/10/15 14:39



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/956,070	10/04/2004	Mai Nguyen	111325-235000	8299

22204 7590 10/24/2005

NIXON PEABODY, LLP
401 9TH STREET, NW
SUITE 900
WASHINGTON, DC 20004-2128

EXAMINER

AUGUSTIN, EVENS J

ART UNIT PAPER NUMBER

3621

DATE MAILED: 10/24/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 10/956,070	Applicant(s) NGUYEN ET AL.	
	Examiner Evans Augustin	Art Unit 3621	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) Responsive to communication(s) filed on 04 October 2004.
- 2a) This action is **FINAL**. 2b) This action is non-final.
- 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) Claim(s) 1-39 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) Claim(s) _____ is/are allowed.
- 6) Claim(s) 1-39 is/are rejected.
- 7) Claim(s) _____ is/are objected to.
- 8) Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) The specification is objected to by the Examiner.
- 10) The drawing(s) filed on 04 October 2004 is/are: a) accepted or b) objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) All b) Some * c) None of:
1. Certified copies of the priority documents have been received.
2. Certified copies of the priority documents have been received in Application No. _____.
3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| 3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date <u>8/02/2005</u> . | 6) <input type="checkbox"/> Other: _____ |

Status of Claims

1. Claims 1-39 have been examined.

Claim Rejections - 35 USC § 102

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) The invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

3. Claims 1-45 are rejected under 35 U.S.C. 102(b) as being anticipated by Downs et al (U.S. 6226618).

As per claims 1-39, Downs et al. disclose an invention that broadly relates to the field of electronic commerce and more particularly to a system and related tools for the secure delivery and rights management of digital assets, such as print media, films, games, and music over global communications networks such as the Internet and the World Wide Web. The invention includes the means and devices to (hardware and software combination) (columns 53, lines 65-67, column 54, lines 1-3) - *Claims 25, 38, 39*. The invention comprising of the following:

- Generating usage rights and derivation of those rights (meta rights). For example, a usage right is the ability to distribute the content or making copies, or the ability to compress the content, or type purchase that can be made. Meta rights include the number of copies that can be made or different compression speed or the owning versus renting the content (columns 59-60) and – *Claims 1, 13, 26*

Art Unit: 3621

- The actual number of copies, compression speed or owning versus rental are state variable as they define the rights (column 59, lines 15-30) - *Claims 2-4, 14-16, 27-29*
- Content Usage Control Layer keeps track of the content's copy/play usage and update the copy/play status (column 19, lines 48-50) – *Claims 5, 17, 30*
- State variables represent a collection of states derived from the usage rights (column 59, lines 15-30) - *Claims 6-8, 18-20, 31-33*
- State variable such as whether to own or rent the content does not material until during the transaction or transfer process. System does not specify to the user what decision to make – *Claims 9-10, 21-22, 34-35*
- Generating licensing with rights (column 7, lines 1-10) – *Claims 11, 23, 36*
- Generating a plurality of rights (column 59, lines 37-67) – *Claims 12, 24, 37*

Conclusion

4. *Examiner has pointed out particular references contained in the prior arts of record in the body of this action for the convenience of the applicant. Although the specified citations are representative of the teachings in the art and are applied to the specific limitations within the individual claim, other passages and figures may apply as well. It is respectfully requested that if the applicant is preparing to respond, to consider fully the entire references as potentially teaching all or part of the claimed invention, as well as the context of the passage as taught by the prior arts or disclosed by the examiner.*

Art Unit: 3621

5. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- **Hsu et al. (US 6947910)** - This invention relates to downloading of media files through a communications network.
- **Stefik et al. (US 5629980)** - The present invention relates to the field of distribution and usage rights enforcement for digitally encoded works.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Evens Augustin whose telephone number is 571-272-6860. The examiner can normally be reached on Monday thru Friday 8 to 5 pm.

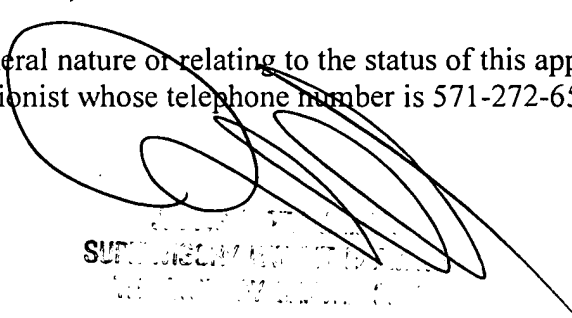
If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jim Trammel can be reached on 571-272-6712.

Any response to this action should be mailed to:

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is 571-272-6584.

Evens J. Augustin
October 17, 2005
Art Unit 3621


SUPERVISOR



Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

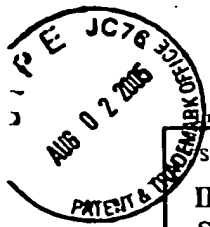
Substitute for form 1449A/PTO		Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>		Application Number	10/956,070
		Filing Date	October 4, 2004
		First Named Inventor	Mai NGUYEN, <i>et al.</i>
		Art Unit	3621
		Examiner Name	Not Yet Assigned
Sheet 1 of 10	Attorney Docket Number	111325-235000	

U.S. PATENT DOCUMENTS						
Examiner Initials ¹	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ² (if known)				
ES		US-3,263,158		07/01/1996	Janis	
		US-3,609,697		09/28/1971	Blevins et al.	
		US-3,790,700		02/05/1974	Callais et al.	
		US-3,798,605		03/19/1974	Feistel	
		US-4,159,468		06/26/1979	Barnes et al.	
		US-4,220,991		09/02/1980	Hamano et al.	
		US-4,278,837		07/14/1981	Best	
		US-4,323,921		04/06/1982	Guillou	
		US-4,442,486		04/10/1984	Mayer	
		US-4,529,870		07/16/1985	Chaum	
		US-4,558,176		12/10/1985	Arnold et al.	
		US-4,593,376		06/03/1986	Volk	
		US-4,614,861		09/30/1986	Pavlov et al.	
		US-4,644,493		02/17/1987	Chandra et al.	
		US-4,658,093		04/14/1987	Hellman	
		US-4,713,753		12/15/1987	Beobert et al.	
		US-4,740,890		04/26/1988	William	
		US-4,796,220		01/03/1989	Wolfe	
		US-4,817,140		03/28/1989	Chandra et al.	
		US-4,827,508		05/02/1989	Shear	
		US-4,868,376		09/19/1989	Lessin et al.	

FOREIGN PATENT DOCUMENTS							
Examiner Initials ¹	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁴
		Country Code ² Number ³	Kind Code ² (if known)				
ES		0 332 304 A3	EP	09/13/1989			
		0 084 441	EP	07/27/1983	TABS LIMITED		
		0 180 460	EP	05/07/1986	SONY CORPORATION		
		0 332 707	EP	09/01/1989	HONDA GIKEN KOGYO KABUSHIKI KAISHA		
		0 651 554	EP	05/03/1995	EASTMAN KODAK CO.		
		0 668 695	EP	08/23/1995	VICTOR COMPANY OF JAPAN LIMITED		
		0 715 244 A	EP	06/05/1996			
		0 715 243 A	EP	06/05/1996			
		0 725 376	EP	08/07/1996	SONY CORP.		
		0 731 404 A1	EP	09/09/1996			
		0 818 748 A2		01/14/1998			

Examiner Signature	<i>[Signature]</i>	Date Considered	10/14/05
--------------------	--------------------	-----------------	----------

¹ EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.
² Applicant's unique citation designation number (optional). ³ See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ⁴ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁵ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁶ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁷ Applicant is to place a check mark here if English language Translation is attached.
 Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.



Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO		<i>Complete if Known</i>	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>		Application Number	10/956,070
		Filing Date	October 4, 2004
		First Named Inventor	Mai NGUYEN, et al.
		Art Unit	3621
		Examiner Name	Not Yet Assigned
Sheet 2 of 10	Attorney Docket Number	111325-235000	

U.S. PATENT DOCUMENTS						
Examiner Initials ¹	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ² (if known)				
EO		US-4,891,838		01/02/1990	Faber	
		US-4,924,378		05/08/1990	Hershey et al.	
		US-4,932,054		06/05/1990	Chou et al.	
		US-4,937,863		06/26/1990	Robert et al.	
		US-4,949,187		08/14/1990	Cohen	
		US-4,953,209		08/28/1990	Ryder, Sr. et al.	
		US-4,961,142		10/02/1990	Elliott et al.	
		US-4,975,647		12/04/1990	Downer et al.	
		US-4,977,594		12/11/1990	Shear	
		US-4,999,806		03/12/1991	Chernow et al.	
		US-5,010,571		04/23/1991	Katznelson	
		US-5,014,234		05/07/1991	Edwards, Jr.	
		US-5,023,907		06/11/1991	Johnson et al.	
		US-5,047,928		09/10/1991	Wiedemer	
		US-5,050,213		09/17/1991	Shear	
		US-5,052,040		09/24/1991	Preston et al.	
		US-5,058,164		10/15/1991	Elmer et al.	
		US-5,103,476		04/07/1992	Waite et al.	
		US-5,113,519		05/12/1992	Johnson et al.	
		US-5,136,643		08/04/1992	Fischer	

FOREIGN PATENT DOCUMENTS							
Examiner Initials ¹	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
		Country Code ³ Number ⁴	Kind Code ² (if known)				
EO		05-268415	JP	10/15/1993	RICOH CO LTD		Abst
		06-175794	JP	06/24/1994	FUJI XEROX CO LTD		Abst
		06-215010	JP	08/05/1994	SONY CORP.		Abst
		07-084852	JP	03/31/1995	HITACHI LTD.		Abst
		07-200317	JP	08/04/1995	TOSHIBA CORP.		Abst
		07-244639	JP	09/19/1995	FUJITSU LTD		Abst
		62-241061	JP	10/21/1987	NEC CORP.		Abst
		64-068835	JP	03/14/1989	RYOICHI MORI		Abst
			WO 00/08909 A		02/24/2000		
			WO 01 13198 A		01/22/2001		
			WO 01/63528 PCT		08/30/2001	IPDN COPR.	
			WO 92/20022 PCT		11/12/1992	DIGITAL EQUIPMENT CORP.	

Examiner Signature	<i>[Signature]</i>	Date Considered	10/14/05
--------------------	--------------------	-----------------	----------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO				<i>Complete if Known</i>	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Application Number	10/956,070
				Filing Date	October 4, 2004
				First Named Inventor	Mai NGUYEN, et al.
				Art Unit	3621
				Examiner Name	Not Yet Assigned
Sheet	4	of	10	Attorney Docket Number	111325-235000

U.S. PATENT DOCUMENTS						
Examiner Initials [*]	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ² (if known)				
↓		US-5,337,357		08/09/1994	Chou et al.	
		US-5,386,369		01/31/1995	Christiano	
		US-5,339,091		08/16/1994	Yamazaki et al.	
		US-5,341,429		08/23/1994	Stringer et al.	
		US-5,347,579		09/13/1994	Blandford	
		US-5,381,526		01/10/1995	Ellson	
		US-5,394,469		02/28/1995	Nagel et al.	
		US-5,410,598		04/25/1995	Shear	
		US-5,412,717		05/02/1995	Fischer	
		US-5,428,606		06/27/1995	Moskowitz	
		US-5,432,849		07/11/1995	Johnson et al.	
		US-5,438,508		08/01/1995	Wyman	
		US-5,444,779		08/22/1995	Daniele	
		US-5,453,601		09/26/1995	Rosen	
		US-5,455,953		10/03/1995	Russell	
		US-5,457,746		10/10/1995	Dolphin	
		US-5,473,687		12/05/1995	Lipscomb et al.	
		US-5,473,692		12/05/1995	Davis	
		US-5,499,298		03/12/1996	Narasimhalu et al.	
		US-5,502,766		03/26/1996	Boebert et al.	
	US-5,504,814		04/02/1996	Miyahara		

FOREIGN PATENT DOCUMENTS							
Examiner Initials [*]	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
		Country Code ³ Number ⁴	Kind Code ² (if known)				
↓		0 715 246 A	EP	06/05/1996			
		1 041 823 A2	EP	10/04/2000			
		2 136 175	GB	09/12/1984	ATALLA CORP.		
		2 236 604	GB	04/10/1991	SUN MICROSYSTEMS INC		
		0 715 241	JP	06/05/1996	MITSUBISHI CORP.		
		04-369068	JP	12/21/1992	CHIYUUBU NIHON DENKI SOFUTOUEA KK		Abst

Examiner Signature	<i>[Signature]</i>	Date Considered	10/14/04
--------------------	--------------------	-----------------	----------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO		<i>Complete if Known</i>	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>		Application Number	10/956,070
		Filing Date	October 4, 2004
		First Named Inventor	Mai NGUYEN, et al.
		Art Unit	3621
		Examiner Name	Not Yet Assigned
Sheet	5	of	10
		Attorney Docket Number	111325-235000

U.S. PATENT DOCUMENTS						
Examiner Initials ¹	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number	Kind Code ² (if known)			
CA		US-5,504,818		04/02/1996	Okano	
		US-5,504,837		04/02/1996	Griffeth et al.	
		US-5,509,070		04/16/1996	Schull	
		US-5,530,235		06/25/1996	Stefik et al.	
		US-5,532,920		07/02/1996	Hartrick et al.	
		US-5,534,975		07/09/1996	Stefik et al.	
		US-5,539,735		07/23/1996	Moskowitz	
		US-5,563,946		10/08/1996	Cooper et al.	
		US-5,568,552		10/22/1996	Davis	
		US-5,621,797		04/15/1997	Rosen	
		US-5,629,980		05/13/1997	Stefik et al.	
		US-5,633,932		05/27/1997	Davis et al.	
		US-5,634,012		05/27/1997	Stefik et al.	
		US-5,638,443		06/10/1997	Stefik et al.	
		US-5,649,013		07/15/1997	Stuckey et al.	
		US-5,655,077		08/05/1997	Jones et al.	
		US-5,671,412		09/1997	Christiano	
		US-5,708,717		01/13/1998	Alasia	
		US-5,734,823		03/31/1998	Saigh et al.	
		US-5,734,891		03/31/1998	Saigh	
	US-5,737,413		04/07/1998	Akiyama et al.		

FOREIGN PATENT DOCUMENTS							
Examiner Initials ¹	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁴
		Country Code ³	Number ³				

Examiner Signature		Date Considered	10/14/05
--------------------	---	-----------------	----------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO				<i>Complete if Known</i>	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Application Number	10/956,070
				Filing Date	October 4, 2004
				First Named Inventor	Mai NGUYEN, et al.
				Art Unit	3621
				Examiner Name	Not Yet Assigned
Sheet	6	of	10	Attorney Docket Number	111325-235000

U.S. PATENT DOCUMENTS						
Examiner Initials ¹	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ² (if known)				
		US-5,737,416		04/07/1998	Cooper et al.	
		US-5,745,569		04/28/1998	Moskowitz et al.	
		US-5,748,783		05/05/1998	Rhoads	
		US-5,757,907		05/26/1998	Cooper et al.	
		US-5,758,069		05/26/1998	Olsen	
		US-5,761,686		06/02/1998	Bloomberg	
		US-5,764,807		06/09/1998	Pearlman et al.	
		US-5,765,152		06/09/1998	Erickson	
		US-5,768,426		06/16/1998	Rhoads	
		US-5,790,664		08/1998	Coley et al.	
		US-5,794,207		08/11/1998	Walker et al.	
		US-5,825,892		10/20/1998	Braudaway et al.	
		US-5,848,154		12/08/1998	Nishio et al.	
		US-5,892,900		04/06/1999	Ginter et al.	
		US-5,910,987		06/08/1999	Ginter et al.	
		US-5,915,019		06/22/1999	Ginter et al.	
		US-5,917,912		06/29/1999	Ginter et al.	
		US-5,920,861		07/06/1999	Hall et al.	
		US-5,925,127		07/1999	Ahmad	
		US-5,940,504		08/17/1999	Griswold	
		US-5,943,422		08/24/1999	Van Wie et al.	
		US-5,949,876		09/07/1999	Ginter et al.	
		US-5,982,891		11/09/1999	Ginter et al.	
		US-5,991,306		11/23/1999	Burns, et al.	
		US-5,999,949		12/07/1999	Crandall	
		US-6,009,401		12/1999	Horstmann	
		US-6,047,067		04/04/2000	Rosen	
		US-6,056,786		05/2000	Rivera et al.	
		US-6,112,181		08/29/2000	Shear et al.	
		US-6,112,239		08/29/2000	Kenner et al.	

FOREIGN PATENT DOCUMENTS							
Examiner Initials ¹	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
		Country Code ³ Number ⁴	Kind Code ² (if known)				

Examiner Signature		Date Considered	10/14/06
-----------------------	---	--------------------	----------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO				<i>Complete if Known</i>	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Application Number	10/956,070
				Filing Date	October 4, 2004
				First Named Inventor	Mai NGUYEN, et al.
				Art Unit	3621
				Examiner Name	Not Yet Assigned
Sheet	7	of	10	Attorney Docket Number	111325-235000

U.S. PATENT DOCUMENTS						
Examiner Initials*	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ² (if known)				
<input checked="" type="checkbox"/>		US-6,115,471		09/05/2000	Oki et al.	
		US-6,138,119		10/24/2000	Hall et al.	
		US-6,157,721		12/05/2000	Shear et al.	
		US-6,169,976 B1		01/02/2001	Colosso	
		US-6,185,683		02/06/2001	Ginter et al.	
		US-6,226,618		05/01/2001	Downs et al.	
		US-6,233,684		05/15/2001	Stefik et al.	
		US-6,236,971 B1		05/22/2001	Stefik et al.	
		US-6,237,786		05/29/2001	Ginter et al.	
		US-6,240,185		05/29/2001	Van Wie et al.	
		US-6,253,193		06/26/2001	Ginter et al.	
		US-6,266,618		05/01/2001	Downs, et al.	
		US-6,292,569		09/18/2001	Shear et al.	
		US-6,301,660		10/09/2001	Benson	
		US-6,327,652		12/04/2001	England et al.	
		US-6,330,670		12/11/2001	England et al.	
		US-6,345,256		02/05/2002	Milsted et al.	
		US-6,363,488		05/26/2002	Ginter et al.	
		US-6,389,402		05/14/2002	Ginter et al.	

FOREIGN PATENT DOCUMENTS							
Examiner Initials*	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T*
		Country Code ³ Number ⁴	Kind Code ² (if known)				

Examiner Signature		Date Considered	10/14/05
--------------------	--	-----------------	----------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.


¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				<i>Complete if Known</i>		
				Application Number	10/956,070	
Sheet		8	of	10	Filing Date	October 4, 2004
					First Named Inventor	Mai NGUYEN, et al.
					Art Unit	3621
					Examiner Name	Not Yet Assigned
					Attorney Docket Number	111325-235000

OTHER PRIOR ART - NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
ES		"National Semiconductor and EPR Partner for Information Metering/Data Security Cards" March 4, 1994, Press Release from Electronic Publishing Resources, Inc.	
		Weber, R., "Digital Rights Management Technology" October 1995	
		Flasche, U. et al., "Decentralized Processing of Documents", pp. 119-131, 1986, Comput. & Graphics, Vol. 10, No. 2	
		Mori, R. et al., "Superdistribution: The Concept and the Architecture", pp. 1133-1146, 1990. The Transactions of the IEICE, Vo. E 73, No. 7, Tokyo, JP	
		Weber, R., "Metering Technologies for Digital Intellectual Property", pp. 1-29, Oct. 1994, A Report to the International Federation of Reproduction Rights Organizations	
		Clark, P.C. et al., "Bits: A Smartcard protected Operating System", pp. 66-70 and 94, November 1994, Communications of the ACM, Vol. 37, No. 11	
		Ross, P.E., "Data Guard", pp. 101, June 6, 1994, Forbes	
		Saigh, W.K., "Knowledge is Sacred", 1992, Video Pocket/Page Reader Systems, Ltd.	
		Kahn, R.E., "Deposit, Registration and Recordation in an Electronic Copyright Management System", pp. 1-19, August 1992, Corporation for National Research Initiatives, Virginia	
		Hilts, P. et al., "Books While U Wait", pp. 48-50, January 3, 1994, Publishers Weekly	
		Stratner, A., "Cash Register on a Chip may Revolutionize Software Pricing and Distribution; Wave Systems Corp.", pp. 1-3, April 1994, Computer Shopper, Vol. 14, No. 4, ISSN 0886-0556	

Examiner Signature		Date Considered	10/14/05
--------------------	---	-----------------	----------

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO			Complete if Known		
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>			Application Number	10/956,070	
			Filing Date	October 4, 2004	
			First Named Inventor	Mai NGUYEN, <i>et al.</i>	
			Art Unit	3621	
			Examiner Name	Not Yet Assigned	
Sheet	9	of	10	Attorney Docket Number	111325-235000

OTHER PRIOR ART - NON PATENT LITERATURE DOCUMENTS			
Examiner Initials ²	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ³
EA		O'Conner, M., "New Distribution Option for Electronic Publishers; iOpener Data Encryption and Metering System for CD-ROM use; Column", pp. 1-6, March 1994, CD-ROM Professional, Vol. 7, No. 2, ISSN: 1409-0833	
		Willett, S., "Metered PCs: Is Your System Watching You? Wave System beta tests new technology", pp. 84, May 2, 1994, InfoWorld	
		Linn, R., "Copyright and Information Services in the Context of the National Research and Education Network", pp. 9-20, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Perrit, Jr., H., "Permission Headers and Contract Law", pp. 27-48, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Upthegrove, L., "Intellectual Property Header Descriptors: A Dynamic Approach", pp. 63-66, January 1994, IMA Intellectual Property Proceedings, Vol. 1, Issue 1	
		Sirbu, M., "Internet Billing Service Design and prototype Implementation", pp. 67-80, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Simmell, S. et al., "Metering and Licensing of Resources: Kala's General Purpose Approach", pp. 81-110, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Kahn, R., "Deposit, Registration and Recordation in an Electronic Copyright Management System", pp. 111-120, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Tygar, J. et al., "Dyad: A System for Using Physically Secure Coprocessors", pp. 121-152, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Griswold, G., "A Method for Protecting Copyright on Networks", pp. 169-178, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	

Examiner Signature		Date Considered	10/14/04
-----------------------	---	--------------------	----------

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				<i>Complete if Known</i>		
				Application Number	10/956,070	
Sheet		10	of	10	Examiner Name	Not Yet Assigned
					Attorney Docket Number	111325-235000
					First Named Inventor	Mai NGUYEN, et al.
					Filing Date	October 4, 2004
					Art Unit	3621

OTHER PRIOR ART - NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
SA		Nelson, T., "A Publishing and Royalty Model for Networked Documents", pp. 257-259, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Robinson, E., "Redefining Mobile Computing", pp. 238-240, 247-248 and 252, July 1993, PC Computing	
		Abadi, M. et al., "Authentication and Delegation with Smart-cards", PP. 1-24, 1990, Research Report DEC Systems Research Center	
		Mark Stefik, "Letting Loose the Light: Igniting Commerce in Electronic Publication", pp. 219-253, 1996, Internet Dreams: Archetypes, Myths, and Metaphors, IDSN 0-262-19373-6	
		Mark Stefik, "Letting Loose the Light: Igniting Commerce in Electronic Publication", pp. 2-35, February 8, 1995, Internet Dreams: Archetypes, Myths and Metaphors	
JK		Henry H. Perritt, Jr., "Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment", April 2-3, 1993, Knowbots, Permissions Headers & Contract Law	

Examiner Signature		Date Considered	10/14/05
--------------------	---	-----------------	----------

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Index of Claims



Application/Control No.

10/956,070

Examiner

Evans Augustin

Applicant(s)/Patent under Reexamination

NGUYEN ET AL.

Art Unit

3621

√	Rejected
=	Allowed

-	(Through numeral) Cancelled
+	Restricted

N	Non-Elected
I	Interference

A	Appeal
O	Objected

Claim		Date			
Final	Original	10/14/05			
	1	√			
	2	√			
	3	√			
	4	√			
	5	√			
	6	√			
	7	√			
	8	√			
	9	√			
	10	√			
	11	√			
	12	√			
	13	√			
	14	√			
	15	√			
	16	√			
	17	√			
	18	√			
	19	√			
	20	√			
	21	√			
	22	√			
	23	√			
	24	√			
	25	√			
	26	√			
	27	√			
	28	√			
	29	√			
	30	√			
	31	√			
	32	√			
	33	√			
	34	√			
	35	√			
	36	√			
	37	√			
	38	√			
	39	√			
	40				
	41				
	42				
	43				
	44				
	45				
	46				
	47				
	48				
	49				
	50				

Claim		Date			
Final	Original				
	51				
	52				
	53				
	54				
	55				
	56				
	57				
	58				
	59				
	60				
	61				
	62				
	63				
	64				
	65				
	66				
	67				
	68				
	69				
	70				
	71				
	72				
	73				
	74				
	75				
	76				
	77				
	78				
	79				
	80				
	81				
	82				
	83				
	84				
	85				
	86				
	87				
	88				
	89				
	90				
	91				
	92				
	93				
	94				
	95				
	96				
	97				
	98				
	99				
	100				

Claim		Date			
Final	Original				
	101				
	102				
	103				
	104				
	105				
	106				
	107				
	108				
	109				
	110				
	111				
	112				
	113				
	114				
	115				
	116				
	117				
	118				
	119				
	120				
	121				
	122				
	123				
	124				
	125				
	126				
	127				
	128				
	129				
	130				
	131				
	132				
	133				
	134				
	135				
	136				
	137				
	138				
	139				
	140				
	141				
	142				
	143				
	144				
	145				
	146				
	147				
	148				
	149				
	150				

Notice of References Cited	Application/Control No. 10/956,070	Applicant(s)/Patent Under Reexamination NGUYEN ET AL.	
	Examiner Evens Augustin	Art Unit 3621	Page 1 of 1

U.S. PATENT DOCUMENTS

*	Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
A	US-5,629,980	05-1997	Stefik et al.	705/54
B	US-6,226,618	05-2001	Downs et al.	705/1
C	US-6,233,684	05-2001	Stefik et al.	713/176
D	US-2003/0066884	04-2003	Reddy et al.	235/382.5
E	US-6,947,910	09-2005	Hsu et al.	705/57
F	US-			
G	US-			
H	US-			
I	US-			
J	US-			
K	US-			
L	US-			
M	US-			

FOREIGN PATENT DOCUMENTS

*	Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
N					
O					
P					
Q					
R					
S					
T					

NON-PATENT DOCUMENTS

*	Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
U	
V	
W	
X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
 United States Patent and Trademark Office
 Address: COMMISSIONER FOR PATENTS
 P.O. Box 1450
 Alexandria, Virginia 22313-1450
 www.uspto.gov



Bib Data Sheet

CONFIRMATION NO. 8299

SERIAL NUMBER 10/956,070	FILING DATE 10/04/2004	CLASS 705	GROUP ART UNIT 3621	ATTORNEY DOCKET NO. 111325-235000
	RULE			

APPLICANTS

Mai Nguyen, Buena Park, CA;
 Xin Wang, Torrance, CA;
 Eddie J. Chen, Rancho Palos Verdes, CA; Bijan Tadayon, Germantown, MD;

** CONTINUING DATA *****

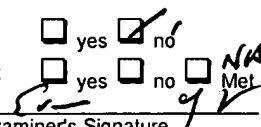
This application is a CIP of 10/162,212 06/05/2002
 which is a CIP of 09/867,745 05/31/2001 PAT 6,754,642



** FOREIGN APPLICATIONS *****

IF REQUIRED, FOREIGN FILING LICENSE GRANTED

** 01/03/2005

Foreign Priority claimed 35 USC 119 (a-d) conditions met	<input type="checkbox"/> yes <input checked="" type="checkbox"/> no <input type="checkbox"/> yes <input type="checkbox"/> no <input checked="" type="checkbox"/> N/A Met after Allowance	STATE OR COUNTRY CA	SHEETS DRAWING 17	TOTAL CLAIMS 39	INDEPENDENT CLAIMS 3
Verified and Acknowledged	Examiner's Signature  Initials SA				

ADDRESS

22204
 NIXON PEABODY, LLP
 401 9TH STREET, NW
 SUITE 900
 WASHINGTON, DC
 20004-2128

TITLE

System and method for rights offering and granting using shared state variables

FILING FEE RECEIVED 1262	FEES: Authority has been given in Paper No. _____ to charge/credit DEPOSIT ACCOUNT No. _____ for following:	<input type="checkbox"/> All Fees <input type="checkbox"/> 1.16 Fees (Filing) <input type="checkbox"/> 1.17 Fees (Processing Ext. of time) <input type="checkbox"/> 1.18 Fees (Issue) <input type="checkbox"/> Other _____
-----------------------------	---	--



TRANSMITTAL FORM <i>(to be used for all correspondence after initial filing)</i>	Application Number	10/956,070
	Filing Date	October 4, 2004
	First Named Inventor	Mai NGUYEN, <i>et al.</i>
	Group Art Unit	3621
	Examiner Name	Augustin Evens
Total Number of Pages in This Submission	Attorney Docket Number	111325-235000

ENCLOSURES <i>(check all that apply)</i>		
<input checked="" type="checkbox"/> Fee Transmittal Form <input type="checkbox"/> Fee Attached <input checked="" type="checkbox"/> Amendment / Reply <input type="checkbox"/> After Final <input type="checkbox"/> Affidavits/declaration(s) <input type="checkbox"/> Extension of Time Request <input type="checkbox"/> Express Abandonment Request <input type="checkbox"/> Information Disclosure Statement <input type="checkbox"/> Certified Copy of Priority Document(s) <input type="checkbox"/> Response to Missing Parts/Incomplete Application <input type="checkbox"/> Response to Missing Parts under 37 CFR 1.52 or 1.53	<input type="checkbox"/> Assignment Papers <i>(for an Application)</i> <input type="checkbox"/> Drawing(s) <input type="checkbox"/> Declaration and Power of Attorney <input type="checkbox"/> Licensing-related Papers <input type="checkbox"/> Petition <input type="checkbox"/> Petition to Convert to a Provisional Application <input type="checkbox"/> Power of Attorney, Revocation Change of Correspondence Address <input type="checkbox"/> Terminal Disclaimer <input type="checkbox"/> Request for Refund <input type="checkbox"/> CD, Number of CD(s) _____	<input type="checkbox"/> After Allowance Communication to Group <input type="checkbox"/> Appeal Communication to Board of Appeals and Interferences <input type="checkbox"/> Appeal Communication to Group <i>(Appeal Notice, Brief, Reply Brief)</i> <input type="checkbox"/> Proprietary Information <input type="checkbox"/> Status Letter <input type="checkbox"/> Application Data Sheet <input type="checkbox"/> Request for Corrected Filing Receipt with Enclosures <input type="checkbox"/> A self-addressed prepaid postcard for acknowledging receipt <input type="checkbox"/> Other Enclosure(s) <i>(please identify below):</i>
Remarks	<input checked="" type="checkbox"/> The Commissioner is hereby authorized to charge any additional fees required or credit any overpayments to Deposit Account No. 19-2380 for the above identified docket number.	

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT	
Firm <i>or</i> Individual name	Carlos R. Villamar Registration No. 43,224 Nixon Peabody LLP 401 9 th Street, N.W., Suite 900 Washington, D.C. 20004-2128
Signature	
Date	January 24, 2006

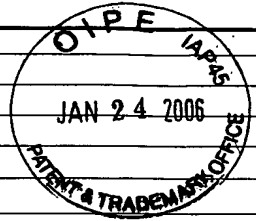
CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]	
I hereby certify that this correspondence is being:	
<input type="checkbox"/> deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop _____, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450	
<input type="checkbox"/> transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) _____.	
_____ Date	_____ Signature
	_____ Typed or printed name

FEE TRANSMITTAL FOR FY 2005

Patent fees are subject to annual revision.

Applicant claims small entity status. See 37 CFR 1.27

<i>Complete if Known</i>	
Application Number	10/956,070
Filing Date	October 4, 2004
First Named Inventor	Mai NGUYEN, et al.
Examiner Name	Augustin Evens
Art Unit	3621
Attorney Docket No.	111325-235000



TOTAL AMOUNT OF PAYMENT **\$200.00**

METHOD OF PAYMENT (check all that apply)

Check Credit Card Money Order Other None

Deposit Account:

Deposit Account Number: **19-2380**

Deposit Account Name: **Nixon Peabody LLP**

The Commissioner is authorized to: (check all that apply)

Charge fee(s) indicated below Credit any overpayments

Charge any additional fee(s)

Charge fee(s) indicated below, except for the filing fee to the above-identified deposit account.

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity		Small Entity		Fee Description
Fee Code	Fee (\$)	Fee Code	Fee (\$)	
1051	130	2051	65	Surcharge - late filing fee or oath
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet
1053	130	1053	130	Non-English specification
1812	2,520	1812	2,520	For filing a request for <i>ex parte</i> reexamination
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action
1251	120	2251	60	Extension for reply within first month
1252	450	2252	225	Extension for reply within second month
1253	1,020	2253	510	Extension for reply within third month
1254	1,590	2254	795	Extension for reply within fourth month
1255	2,160	2255	1,080	Extension for reply within fifth month
1401	500	2401	250	Notice of Appeal
1402	500	2402	250	Filing a brief in support of an appeal
1403	1,000	2403	500	Request for oral hearing
1451	1,510	1451	1,510	Petition to institute a public use proceeding
1452	500	2452	250	Petition to revive - unavoidable
1453	1,500	2453	750	Petition to revive - unintentional
1501	1,400	2501	700	Utility issue fee (or reissue)
1502	800	2502	400	Design issue fee
1503	1,100	2503	550	Plant issue fee
1460	130	1460	130	Petitions to the Commissioner
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)
1806	180	1806	180	Submission of Information Disclosure Stmt
8021	40	8021	40	Recording each patent assignment per property (times number of properties)
1809	790	2809	395	Filing a submission after final rejection (37 CFR 1.129(a))
1810	790	2810	395	For each additional invention to be examined (37 CFR 1.129(b))
1801	790	2801	395	Request for Continued Examination (RCE)
1802	900	1802	900	Request for expedited examination of a design application

Other fee (specify) _____

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) (\$ 0)

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee (\$)	Small Entity Fee Code	Small Entity Fee (\$)	Fee Description	Fee Paid
1001	300	2001	150	Utility filing fee	
1002	200	2002	100	Design filing fee	
1003	200	2003	100	Plant filing fee	
1004	300	2004	150	Reissue filing fee	
1005	200	2005	100	Provisional filing fee	
SUBTOTAL (1)					(\$ 0)

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

Total Claims	Extra Claims	Fee from below	Fee Paid
43	-39** = 4	50.00	\$200.00
Independent Claims: 3	-3** = 0		0
Multiple Dependent			0

Large Entity Fee Code	Large Entity Fee (\$)	Small Entity Fee Code	Small Entity Fee (\$)	Fee Description
1202	50	2202	25	Claims in excess of 20
1201	200	2201	100	Independent claims in excess of 3
1203	360	2203	180	Multiple dependent claim, if not paid
1204	200	2204	100	** Reissue independent claims over original patent
1205	50	2205	25	** Reissue claims in excess of 20 and over original patent
SUBTOTAL (2) \$200.00				

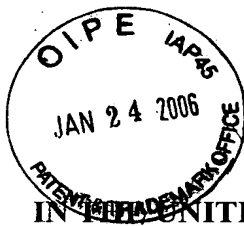
**or number previously paid, if greater; For Reissues, see above

SUBMITTED BY

Name (Print/Type)	Carlos R. Villamar	Registration No. (Attorney/Agent)	43,224	Telephone	(202) 585-8204
Signature		Date	January 24, 2006		

Complete (if applicable)

SEND TO: Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450



Docket No. 111325-235000
Serial No. 10/956,070

Handwritten initials and a symbol.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:) Confirmation No.: 8299
Mai NGUYEN, *et al.*) Group Art Unit: 3621
Serial No. 10/956,070) Examiner: Augustin Evens
Filed: October 4, 2004)
For: **SYSTEM AND METHOD FOR**)
RIGHTS OFFERING AND)
GRANTING USING SHARED STATE)
VARIABLES)

U.S. Patent and Trademark Office
Customer Window, Mail Stop Non-Fee Amendment
Randolph Building
401 Dulany Street
Alexandria, VA 22314

AMENDMENT

Sir:

In response to the non-final Office Action mailed **October 24, 2005**, the below amendments and following remarks are submitted in connection with the above-identified application.

01/25/2006 SZEWDIE1 00000135 192380 10956070
01 FC:1202 200.00 DA

REMARKS

The following amendments and remarks are submitted to be fully responsive to the non-final Official Action of **October 24, 2005**. In the present response, claims 2-10, 14-22, 25, and 27-35 are amended, claims 1, 11-13, 23-24, 26, and 36-39 are cancelled, and claims 40-54 are added. No new matter is introduced. Thus, claims 2-10, 14-22, 25, 27-35, and 40-54 are now pending. Reconsideration and allowance of this application are respectfully requested.

Referring now to the present Office Action, claims 1-39 were rejected under 35 U.S.C. §102(b) as being anticipated by U.S. Patent No. 6,226,618 to *Downs et al.* However, claims 2-10, 14-22, 25, 27-35, and 40-54 are patentably distinguishable over *Downs et al.*, because *Downs et al.* fails to disclose, teach or suggest all of the features recited in the present claims. For example, new independent claim 40 (emphasis added) recites:

A method for sharing rights adapted to be associated with an item, the method comprising:
 specifying in a first license at least one usage right and/or at least one meta-right for the item,
 wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices;
 defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item;
 defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item;
 associating at least one state variable with the at least one right in the first license and that is shared among the one or more users or devices,
 wherein the at least one state variable is used to determine how the shared right is further generated in a second license;
 generating in the second license one or more rights from the usage right and/or the meta-right in the first license,
 wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices;
 associating at least one state variable with the at least one right that is shared in the second license,
 wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

new independent claim 41 (emphasis added) recites:

A system for sharing rights adapted to be associated with an item, the system comprising:
 means for **specifying in a first license at least one usage right and/or at least one meta-right for the item,**
 wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices;

means for defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item;
means for defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item;
means for associating at least one state variable with the at least one right in the first license and that is shared among the one or more users or devices,
wherein the at least one state variable is used to determine how the shared right is further generated in a second license;
means for generating in the second license one or more rights from the usage right and/or the meta-right in the first license,
wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices;
means for associating at least one state variable with the at least one right that is shared in the second license,
wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license; and

new independent claim 42 (emphasis added) recites:

A device for sharing rights adapted to be associated with an item, the device comprising:
means for receiving a first license specifying at least one usage right and/or at least one meta-right for the item,
wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices,
the least one usage right defines a manner of use selected from a plurality of permitted manners of use for the item,
the at least one meta-right defines a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item,
at least one state variable is associated with the at least one right in the first license and is shared among the one or more users or devices,
the at least one state variable is used to determine how the shared right is further generated in a second license; and
means for generating in the second license one or more rights from the usage right and/or the meta-right in the first license,
wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices,
at least one state variable is associated with the at least one right that is shared in the second license, and
the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

By contrast, *Downs et al.* is directed to a method and apparatus of securely providing data to a user's system, wherein the data is encrypted so as to only be decryptable by a data decrypting key, the data decrypting key being encrypted using a first public key, and the encrypted data being accessible to the user's system. The method includes transferring the encrypted data decrypting key to a clearing house that possesses a first private key, which

corresponds to the first public key; decrypting the data decrypting key using the first private key; re-encrypting the data decrypting key using a second public key; transferring the re-encrypted data decrypting key to the user's system, the user's system possessing a second private key, which corresponds to the second public key; and decrypting the re-encrypted data decrypting key using the second private key. However, *Downs et al.* fails to disclose, teach or suggest the noted features recited in independent claims 40, 41 and 42.

For example, *Downs et al.* fails to disclose, teach or suggest meta-rights in the manner claimed and which are rights about rights, such as the right for distributors to issue certain rights to a consumer. By contrast, usage rights are rights for content, such as the right to play or to copy content. The invention recited in independent claims 40, 41 and 42 is not directed to generating rights to use content, including making copies, etc., but rather is directed to rights to derive rights for content. For example, with invention recited in independent claims 40, 41 and 42, a user can be permitted to play content on a PC, to make a copy for a PDA, and to issue rights to play the copy on the PDA. When the user transfers the copy to her PDA, the user also issues to the PDA rights to play the copy and to transfer the issued rights along with the copy. Without the issued rights, the user cannot play the content on the PDA.

By contrast, *Downs et al.* discloses specifying allowed states (e.g., number of copies, compression speed) in Usage Conditions and it is up to the Content Usage Control Layer to keep track of the content's copy/play usage and update the copy/play status. For example, if a Usage Condition specifies a max count of 3 plays, the Content Usage Control Layer may update the number of times the content has been played to ensure that only 3 plays are allowed. This concept, however, does not teach or suggest the noted features recited in independent claims 40, 41 and 42.

In addition, a state variable is not equivalent to a max count or a compression rate. For example, a max count is a constant number, which can be 3 or 5, etc., and a compression rate is another constant number, which can be 384 Kbps or 56 Kbps, etc. By contrast, a state variable can be represented by an identifier and whose values can vary over time. A state variable in the specification of a condition can be used for rights sharing, and which is also feature that differentiates invention recited in independent claims 40, 41 and 42 over *Downs et al.* For example, a content provider can decide how a content is shared among a group of consumers using a state variable and *Downs et al.* is also deficient in this respect.

The inventions recited in independent claims 40, 41 and 42 and claims dependent therefrom recognize and solve the following problems:

[0009] However, there are limitations associated with the above-mentioned paradigms wherein only usage rights and conditions associated with content are specified by content owners or other grantors of rights. Once purchased by an end user, a consumer, or a distributor, of content along with its associated usage rights and conditions has no means to be legally passed on to a next recipient in a distribution chain. Further the associated usage rights have no provision for specifying rights to derive other rights, i.e. Rights to modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, revoke, or the like. Common content distribution models often include a multi-tier distribution and usage chain. Known DRM systems do not facilitate the ability to prescribe rights and conditions for all participants along a content distribution and usage chain. Therefore, it is difficult for a content owner to commercially exploit content unless the owner has a relationship with each party in the distribution chain.

The inventions recited in independent claims 40, 41 and 42 and claims dependent therefrom provide the following advantages:

[0090] There are multiple ways to specify the scope of state variables, each of which can affect whether the derivative state variables can be shared, how the derivative state variables can be shared, and the like. For example, a state variable can be local, and solely confined to a recipient or can be global, and shared by a predetermined group of recipients. A global state variable can be shared by a group of recipients not determined when derived rights are issued, but to be specified later, perhaps based on certain rules defined in the license or based on other means. A global state variable can be shared between one or more rights suppliers, predetermined recipients, un-specified recipients, and the like. Advantageously, depending on the sharing employed with a given a business model and the rights granted in the meta-rights, state variables can be created at different stages of the value chain.

By contrast, *Downs et al.* fails disclose, teach or suggest the noted features, fails to recognize or solve the noted problems, and fails to provide the advantages of the inventions recited in independent claims 40, 41 and 42.

The dependent claims are allowable on their on merits and for at least the reasons as argued above with respect to independent claims 40, 41 and 42.

The references that have been cited, but not applied by the Examiner, have been taken into consideration during formulation of this response. However, since such references were not considered by the Examiner to be of sufficient relevance to apply against any of the claims, no detailed comments thereon are believed to be warranted at this time.

In view of the foregoing, it is submitted that the present application is in condition for allowance and a notice to that effect is respectfully requested. However, if the Examiner deems that any issue remains after considering this response, the Examiner is invited to

contact the undersigned attorney to expedite the prosecution and engage in a joint effort to work out a mutually satisfactory solution.

Respectfully submitted,

NIXON PEABODY, LLP

/Carlos R. Villamar, Reg. # 43,224/

Carlos R. Villamar

Reg. No. 43,224

NIXON PEABODY LLP
CUSTOMER NO.: 22204
401 9th Street, N.W., Suite 900
Washington, DC 20004
Tel: 202-585-8000
Fax: 202-585-8080

Amendments to the Claims:

1. (Cancelled)
2. (Currently amended) The method of ~~claim 1~~ claim 40, wherein the state variable in the first or second license inherits a state thereof for content usage or rights ~~transfer~~ derivation from other generated usage rights and meta-rights.
3. (Currently amended) The method of ~~claim 1~~ claim 40, wherein the state variable in the first or second license shares a state thereof for content usage or rights ~~transfer~~ derivation with other generated usage rights and meta-rights.
4. (Currently amended) The method of ~~claim 1~~ claim 40, wherein the state variable in the first or second license inherits a remaining state for content usage or rights ~~transfer~~ derivation from other generated usage rights and meta-rights.
5. (Currently amended) The method of ~~claim 1~~ claim 40, wherein the state variable in the first or second license is updated upon exercise of a right associated with the state variable.
6. (Currently amended) The method of ~~claim 1~~ claim 40, wherein the state variable in the first or second license represents a collection of states.
7. (Currently amended) The method of ~~claim 1~~ claim 40, ~~further comprising deriving at least one right from the generated usage rights and meta-rights, wherein the derived right includes at least one state variable that is shared with or inherited from the generated usage rights and meta-rights and is used for determining a state of the derived right~~ further comprising:
generating in a third license one or more rights from the usage right and/or the meta-right in the second license,
wherein the one or more rights in the third license includes at least one right that is shared among one or more users or devices;

associating at least one state variable with the at least one right that is shared in the third license.

wherein the at least one state variable that is associated with the third license is based on the at least one state variable that is associated with the second license.

8. (Currently amended) The method of ~~claim 7~~ claim 40, further comprising a plurality of state variables that determine the state of the ~~derived~~ at least one right that is shared in the first or the second license.

9. (Currently amended) The method of ~~claim 7~~ claim 40, wherein the state variable is unspecified in the ~~derived~~ at least one right that is shared in the first license, is created during a rights ~~transfer~~ derivation, and is ~~assigned~~ associated to the ~~derived~~ at least one right that is shared in the second license.

10. (Currently amended) The method of ~~claim 7~~ claim 40, wherein the state variable in the second license is transferred from the ~~generated~~ usage rights and meta-rights in the first license and is associated with [[to]] the ~~derived~~ right that is shared in the second license.

11-13. (Cancelled)

14. (Currently amended) The system of ~~claim 13~~ claim 41, wherein the state variable in the first or second license inherits a state thereof for content usage or rights ~~transfer~~ derivation from other generated usage rights and meta-rights.

15. (Currently amended) The system of ~~claim 13~~ claim 41, wherein the state variable in the first or second license shares a state thereof for content usage or rights derivation with other generated usage rights and meta-rights.

16. (Currently amended) The system of ~~claim 13~~ claim 41, wherein the state variable in the first or second license inherits a remaining state for content usage or rights ~~transfer~~ derivation from other generated usage rights and meta-rights.

17. (Currently amended) The system of ~~claim 13~~ claim 41, wherein the state variable in the first or second license is updated upon exercise of a right associated with the state variable.

18. (Currently amended) The system of ~~claim 13~~ claim 41, wherein the state variable in the first or second license represents a collection of states.

19. (Currently amended) The system of ~~claim 13~~ claim 41, ~~further comprising deriving at least one right from the generated usage rights and meta rights, wherein the derived right includes at least one state variable that is shared with or inherited from the generated usage rights and meta rights and is used for determining a state of the derived right~~ further comprising:

means for generating in a third license one or more rights from the usage right and/or the meta-right in the second license,

wherein the one or more rights in the third license includes at least one right that is shared among one or more users or devices;

means for associating at least one state variable with the at least one right that is shared in the third license,

wherein the at least one state variable that is associated with the third license is based on the at least one state variable that is associated with the second license.

20. (Currently amended) The system of ~~claim 19~~ claim 41, including a plurality of state variables that determine the state of the ~~derived~~ at least one right that is shared in the first or the second license.

21. (Currently amended) The system of ~~claim 19~~ claim 41, wherein the state variable is unspecified in the ~~derived~~ at least one right that is shared in the first license, is created during a rights ~~transfer~~ derivation, and is ~~assigned~~ associated to the ~~derived~~ at least one right that is shared in the second license.

22. (Currently amended) The system of ~~claim 19~~ claim 41, wherein the state variable in the second license is transferred from the ~~generated~~ usage rights and meta-rights in the first license and is associated with [[to]] the derived right that is shared in the second license.

23-24. (Cancelled)

25. (Currently amended) The system of ~~claim 13~~ claim 41, wherein the means for generating; the means for defining via the usage rights, and the means for defining via the meta rights ~~comprise at least one of computer executable instructions, and devices of a computer system~~ is implemented with one or more hardware and/or software components.

26. (Cancelled)

27. (Currently amended) The device of ~~claim 26~~ claim 42, wherein the state variable in the first or second license inherits a state thereof for content usage or rights ~~transfer derivation~~ from other generated usage rights and meta-rights.

28. (Currently amended) The device of ~~claim 26~~ claim 42, wherein the state variable in the first or second license shares a state thereof for content usage or rights ~~transfer derivation~~ with other generated usage rights and meta-rights.

29. (Currently amended) The device of ~~claim 26~~ claim 42, wherein the state variable in the first or second license inherits a remaining state for content usage or rights ~~transfer derivation~~ from other generated usage rights and meta-rights.

30. (Currently amended) The device of ~~claim 26~~ claim 42, wherein the state variable in the first or second license is updated upon exercise of a right associated with the state variable.

31. (Currently amended) The device of ~~claim 26~~ claim 42, wherein the state variable in the first or second license represents a collection of states.

32. (Currently amended) The device of ~~claim 26~~ claim 42, ~~further comprising deriving at least one right from the generated usage rights and meta rights, wherein the derived right includes at least one state variable that is shared with or inherited from the generated usage rights and meta rights and is used for determining a state of the derived right~~ wherein a third

license includes one or more rights from the usage right and/or the meta-right in the second license,

the one or more rights in the third license includes at least one right that is shared among one or more users or devices,

at least one state variable is associated with the at least one right that is shared in the third license, and

the at least one state variable that is associated with the third license is based on the at least one state variable that is associated with the second license.

33. (Currently amended) The device of ~~claim 32~~ claim 42, including a plurality of state variables that determine the state of the ~~derived~~ at least one right that is shared in the first or the second license.

34. (Currently amended) The device of ~~claim 32~~ claim 42, wherein the state variable is unspecified in the ~~derived~~ at least one right that is shared in the first license, is created during a rights ~~transfer~~ derivation, and is ~~assigned~~ associated to the ~~derived~~ at least one right that is shared in the second license.

35. (Currently amended) The device of ~~claim 32~~ claim 42, wherein the state variable in the second license is transferred from the ~~generated~~ usage rights and meta-rights in the first license and is associated with [[to]] the derived right that is shared in the second license.

36-39. (Cancelled)

40. (New) A method for sharing rights adapted to be associated with an item, the method comprising:

specifying in a first license at least one usage right and/or at least one meta-right for the item,

wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices;

defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item;

defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item;

associating at least one state variable with the at least one right in the first license and that is shared among the one or more users or devices,

wherein the at least one state variable is used to determine how the shared right is further generated in a second license;

generating in the second license one or more rights from the usage right and/or the meta-right in the first license,

wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices;

associating at least one state variable with the at least one right that is shared in the second license,

wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

41. (New) A system for sharing rights adapted to be associated with an item, the system comprising:

means for specifying in a first license at least one usage right and/or at least one meta-right for the item,

wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices;

means for defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item;

means for defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item;

means for associating at least one state variable with the at least one right in the first license and that is shared among the one or more users or devices,

wherein the at least one state variable is used to determine how the shared right is further generated in a second license;

means for generating in the second license one or more rights from the usage right and/or the meta-right in the first license,

wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices;

means for associating at least one state variable with the at least one right that is shared in the second license,

wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

42. (New) A device for sharing rights adapted to be associated with an item, the device comprising:

means for receiving a first license specifying at least one usage right and/or at least one meta-right for the item,

wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices,

the least one usage right defines a manner of use selected from a plurality of permitted manners of use for the item,

the at least one meta-right defines a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item,

at least one state variable is associated with the at least one right in the first license and is shared among the one or more users or devices,

the at least one state variable is used to determine how the shared right is further generated in a second license; and

means for generating in the second license one or more rights from the usage right and/or the meta-right in the first license,

wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices,

at least one state variable is associated with the at least one right that is shared in the second license, and

the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

43. (New) The method of claim 40, wherein the method is implemented with one or more hardware and/or software components configured to perform the steps of the method.

44. (New) The method of claim 40, wherein the method is implemented with one or more computer readable instructions embedded on a computer readable medium and configured to cause one or more computer processors to perform the steps of the method.

45. (New) The device of claim 42, wherein the device is implemented with one or more hardware and/or software components.

46. (New) The method of claim 40, wherein the specifying step includes specifying in the first license at least one usage right and at least one meta-right for the item.

47. (New) The system of claim 41, wherein the specifying means includes means for specifying in the first license at least one usage right and at least one meta-right for the item.

48. (New) The device of claim 42, wherein the first license specifies at least one usage right and at least one meta-right for the item.

49. (New) The method of claim 40, wherein the plurality of permitted manners of use for the item include copy, transfer, loan, play, print, back-up, restore, delete, extract, embed, edit, authorize, install, and un-install the item.

50. (New) The system of claim 41, wherein the plurality of permitted manners of use for the item include copy, transfer, loan, play, print, back-up, restore, delete, extract, embed, edit, authorize, install, and un-install the item.

51. (New) The device of claim 42, wherein the plurality of permitted manners of use for the item include copy, transfer, loan, play, print, back-up, restore, delete, extract, embed, edit, authorize, install, and un-install the item.

52. (New) The method of claim 40, wherein the plurality of permitted manners of rights derivation for the item include issue, modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, and revoke rights for the item.

53. (New) The system of claim 41, wherein the plurality of permitted manners of rights derivation for the item include issue, modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, and revoke rights for the item.

54. (New) The device of claim 42, wherein the plurality of permitted manners of rights derivation for the item include issue, modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, and revoke rights for the item.

PATENT APPLICATION FEE DETERMINATION RECORD
Effective October 1, 2004

Application or Docket Number

10956070

CLAIMS AS FILED - PART I

	(Column 1)	(Column 2)
TOTAL CLAIMS	39	
FOR	NUMBER FILED	NUMBER EXTRA
TOTAL CHARGEABLE CLAIMS	39 minus 20 =	* 19
INDEPENDENT CLAIMS	3 minus 3 =	* 0
MULTIPLE DEPENDENT CLAIM PRESENT <input type="checkbox"/>		

* If the difference in column 1 is less than zero, enter "0" in column 2

CLAIMS AS AMENDED - PART II

124/06

	(Column 1)	(Column 2)	(Column 3)
AMENDMENT A	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total * 43	Minus ** 39	= 4
	Independent * 3	Minus *** 3	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>			

	(Column 1)	(Column 2)	(Column 3)
AMENDMENT B	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total *	Minus **	=
	Independent *	Minus ***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>			

	(Column 1)	(Column 2)	(Column 3)
AMENDMENT C	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total *	Minus **	=
	Independent *	Minus ***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>			

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20."

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3."

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

SMALL ENTITY TYPE

OR OTHER THAN SMALL ENTITY

RATE	FEE
BASIC FEE	395.00
X\$ 9=	
X44=	
+150=	
TOTAL	

RATE	FEE
BASIC FEE	790.00
X\$18=	342.00
X88=	
+300=	
TOTAL	1132.00

SMALL ENTITY TYPE

OR OTHER THAN SMALL ENTITY

RATE	ADDITIONAL FEE
X\$ 9=	
X44=	
+150=	
TOTAL ADDIT. FEE	

RATE	ADDITIONAL FEE
X\$18=	200.00
X88=	
+300=	
TOTAL ADDIT. FEE	200.00

RATE	ADDITIONAL FEE
X\$ 9=	
X44=	
+150=	
TOTAL ADDIT. FEE	

RATE	ADDITIONAL FEE
X\$18=	
X88=	
+300=	
TOTAL ADDIT. FEE	

RATE	ADDITIONAL FEE
X\$ 9=	
X44=	
+150=	
TOTAL ADDIT. FEE	

RATE	ADDITIONAL FEE
X\$18=	
X88=	
+300=	
TOTAL ADDIT. FEE	



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/956,070	10/04/2004	Mai Nguyen	11 1325-235000	8299

22204 7590 04/17/2006

NIXON PEABODY, LLP
401 9TH STREET, NW
SUITE 900
WASHINGTON, DC 20004-2128

EXAMINER

AUGUSTIN, EVENS J

ART UNIT	PAPER NUMBER
3621	

3621

DATE MAILED: 04/17/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Response to Amendment

This is in response to an amendment file on 1/24/2006 for letter for patent filed on 10/04/2004. In the amendment, claims 1, 12, 16 and 25 have been amended. Claims 1, 11-13, 23-24, 26, and 36-39 have been cancelled. Claims 40-54 have been added. Claims 2-10, 14-22, 25, 27-35, and 40-54 are pending in the letter.

Response to Arguments

1. The United States Patent and Trademark Office has fully considered the applicant's arguments on 1/24/2006, but has not found those arguments to be persuasive.

The claims, as interpreted in light of the specification, are regarding content distribution system, in which the content owners specify the rights of distributors/end-users vis-à-vis the content. The present invention categorizes the rights associated with the content into **usage rights** and **meta-rights**. Usage rights are defined as a manner of use for the items, and meta-rights are defined as a manner of rights transfer for the items (page 4, par. 10). According to the applicant's specification, **meta-rights** can include rights to offer, grant, obtain, transfer, delegate, track, surrender, exchange, and revoke usage rights to/from others (page 9, par. 41). A **usage right** can be the right to print content **three times**. Each time the usage right is exercised, the value of the state variable "number of prints" is incremented. In this example, when the value of the state variable is **three**, the condition is no longer satisfied and content cannot be printed (page 9, par. 43). The content owner initially grants the distributor a license/rights to

Art Unit: 3621

distribute/sell the content, and the distributor can modify the initial licensing/rights from the owner to offer a more customized version of the rights to the end users (pages 24-25, par. 79-80).

Argument 1: Prior Art does not teach the aspects Meta-rights

Response 1: Owners setting initial usage rights/licensing (first license) for content to the distributors (column 21, lines 30-33). Those usage rights can be modified by the digital store (column 21, lines 33-39) to create secondary licensing or customized licensing (column 10, lines 15-18) to the end user

The system also defines the manner in which the content can be used (meta-rights) such as onto what kinds of media the content can be transferred to (column 59, lines 52-54)

Application stands finally rejected.

Status of Claims

2. Claims 2-10, 14-22, 25, 27-35, and 40-54 have been examined.

Claim Rejections - 35 USC § 112

3. Regarding claims 40-43, 45, 7, 19, 25, and 32, the phrase "and/or" renders the claims indefinite because there are uncertainties or ambiguities with respect to the question of scope or clarity of the claims. The claims are rejected under 35 U.S.C. 112, second paragraph (See MPEP § 2173.05(h)).

4. Claims 7, 19 and 32 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claims contain subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention. The claims refer to a third license being generating from a second license. The **specification is silent** with regard to the generation of third license and the rights associated with that third license.

5. Claims 49-53 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claims contain subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention. The claims refer to a third license being generating from a second license. According to the applicant's specification, **meta-rights** can include rights to offer, grant, obtain, transfer, delegate, track, surrender, exchange, and revoke usage rights to/from others (page 9, par. 41). The **specification is silent** with regard to rights such as restore, back up, exercise and extract.

Art Unit: 3621

Claim Rejections - 35 USC § 102

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) The invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

7. Claims 2-10, 14-22, 25, 27-35, and 40-54 are rejected under 35 U.S.C. 102(b) as being anticipated by Downs et al (U.S 6226618).

As per claims 2-10, 14-22, 25, 27-35, and 40-54, Downs et al. disclose an invention that broadly relates to the field of electronic commerce and more particularly to a system and related tools for the secure delivery and rights management of digital assets, such as print media, films, games, and music over global communications networks such as the Internet and the World Wide Web. The invention includes the means and devices to (hardware and software combination) (columns 53, lines 65-67, column 54, lines 1-3) - *Claims 25, 38, 39, 41-45, 48, 51,*

54. The invention comprising of the following:

- Owners setting initial usage rights/licensing (**first license**) for content to the distributors (column 21, lines 30-33) – *Claims 40-42*
- Example of usage rights include (column 59, lines 38-69 – *Claims 40, 49-54:*

Compressed Version	384 Kbps
Type of user	Private Consumer
Type of Transaction	Purchase or Rental
Number Of copies	1
Rental Terms	14 Days

Art Unit: 3621

Transfer on What Media	Mini Disc or Computer
------------------------	-----------------------

- Owners setting initial usage rights/licensing (**first license**) for content to the distributors (column 21, lines 30-33). Those usage rights can be modified by the digital store (column 21, lines 33-39) to create **secondary licensing** or customized licensing (column 10, lines 15-18) to the end user – *Claims 40-42, 10, 22, 35*
- The system also defines the manner in which the content can be used (**meta-rights**) such as onto what kinds of media the content can be transferred to (column 59, lines 52-54) – *Claims 40-42, 46-48*
- State variable such the number of copies a user is allowed to make (column 59, line 50 or rental terms (column 59, lines 55-60) - *Claims 40-42*
- The secondary licensing such as restrictions on rental time period can not violate the initial time period set by the initial licensing (column 21, line 35) - *Claims 40-42*
- The state variable is derived from the usage rights (column 59, line 50) or rental terms (column 59, lines 55-60) - *Claims 2-4, 14-16, 27-29*
- The system keeps track of the content's copy/play usage and updates the copy/play status (column 20, lines 49-50) – *Claims 5, 17, 30*
- A state variable can represent various other states, for example an item that rented can affect the number of copies that can be made or whether or not copies can be made - *Claims 6, 8, 18, 20, 31,33*
- The system embeds a code on every copy the content, as it is transferred form user device to the next. When the Digital Content is accessed in a compliant End-User Devices, the

Art Unit: 3621

End-User Player Application reads the watermark to check the use restrictions and updates the watermark as required. If the requested use of the content does not comply with the usage conditions, e.g., the number of copies has been exhausted, the End-User Device(s) will not perform the request (column 7, lines 40-55) - *Claims 7, 19, 32*

- The content does not specify how the initial set of rights and variable are to modified, as long as it does not violate the initial licensing (column 21, line 35) - *Claims 9, 21, 34*

Conclusion

8. **THIS ACTION IS MADE FINAL.** Any new grounds of rejection is due to the applicant's amendment. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Art Unit: 3621

9. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Evens Augustin whose telephone number is 571-272-6860. The examiner can normally be reached on Monday thru Friday 8 to 5 pm.


If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jim Trammel can be reached on 571-272-6712.

Any response to this action should be mailed to:

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is 571-272-6584.

Evens J. Augustin
April 11, 2006
Art Unit 3621


SALVATORE CANGIALOSI
PRIMARY EXAMINER
ART UNIT 222

Index of Claims



Application/Control No.

10/956,070

Examiner

Evens Augustin

Applicant(s)/Patent under Reexamination

NGUYEN ET AL.

Art Unit

3621

√	Rejected
=	Allowed

-	(Through numeral) Cancelled
+	Restricted

N	Non-Elected
I	Interference

A	Appeal
O	Objected

Claim		Date			
Final	Original	4/11/06			
	1	-			
	2	√			
	3	√			
	4	√			
	5	√			
	6	√			
	7	√			
	8	√			
	9	√			
	10	√			
	11	-			
	12	-			
	13	-			
	14	√			
	15	√			
	16	√			
	17	√			
	18	√			
	19	√			
	20	√			
	21	√			
	22	√			
	23	-			
	24	-			
	25	√			
	26	-			
	27	√			
	28	√			
	29	√			
	30	√			
	31	√			
	32	√			
	33	√			
	34	√			
	35	√			
	36	-			
	37	-			
	38	-			
	39	-			
	40	√			
	41	√			
	42	√			
	43	√			
	44	√			
	45	√			
	46	√			
	47	√			
	48	√			
	49	√			
	50	√			

Claim		Date			
Final	Original	4/11/06			
	51	√			
	52	√			
	53	√			
	54	√			
	55				
	56				
	57				
	58				
	59				
	60				
	61				
	62				
	63				
	64				
	65				
	66				
	67				
	68				
	69				
	70				
	71				
	72				
	73				
	74				
	75				
	76				
	77				
	78				
	79				
	80				
	81				
	82				
	83				
	84				
	85				
	86				
	87				
	88				
	89				
	90				
	91				
	92				
	93				
	94				
	95				
	96				
	97				
	98				
	99				
	100				

Claim		Date			
Final	Original				
	101				
	102				
	103				
	104				
	105				
	106				
	107				
	108				
	109				
	110				
	111				
	112				
	113				
	114				
	115				
	116				
	117				
	118				
	119				
	120				
	121				
	122				
	123				
	124				
	125				
	126				
	127				
	128				
	129				
	130				
	131				
	132				
	133				
	134				
	135				
	136				
	137				
	138				
	139				
	140				
	141				
	142				
	143				
	144				
	145				
	146				
	147				
	148				
	149				
	150				



AP
JFV

TRANSMITTAL FORM <i>(to be used for all correspondence after initial filing)</i>	Application Number	10/956,070
	Filing Date	October 4, 2004
	First Named Inventor	Mai NGUYEN, <i>et al.</i>
	Group Art Unit	3621
	Examiner Name	Augustin Evens
Total Number of Pages in This Submission	Attorney Docket Number	111325-235000

ENCLOSURES <i>(check all that apply)</i>		
<input type="checkbox"/> Fee Transmittal Form <input type="checkbox"/> Fee Attached <input checked="" type="checkbox"/> Amendment / Reply <input checked="" type="checkbox"/> After Final <input type="checkbox"/> Affidavits/declaration(s) <input type="checkbox"/> Extension of Time Request <input type="checkbox"/> Express Abandonment Request <input type="checkbox"/> Information Disclosure Statement <input type="checkbox"/> Certified Copy of Priority Document(s) <input type="checkbox"/> Response to Missing Parts/Incomplete Application <input type="checkbox"/> Response to Missing Parts under 37 CFR 1.52 or 1.53	<input type="checkbox"/> Assignment Papers <i>(for an Application)</i> <input type="checkbox"/> Drawing(s) <input type="checkbox"/> Declaration and Power of Attorney <input type="checkbox"/> Licensing-related Papers <input type="checkbox"/> Petition <input type="checkbox"/> Petition to Convert to a Provisional Application <input type="checkbox"/> Power of Attorney, Revocation Change of Correspondence Address <input type="checkbox"/> Terminal Disclaimer <input type="checkbox"/> Request for Refund <input type="checkbox"/> CD, Number of CD(s) _____	<input type="checkbox"/> After Allowance Communication to Group <input type="checkbox"/> Appeal Communication to Board of Appeals and Interferences <input type="checkbox"/> Appeal Communication to Group <i>(Appeal Notice, Brief, Reply Brief)</i> <input type="checkbox"/> Proprietary Information <input type="checkbox"/> Status Letter <input type="checkbox"/> Application Data Sheet <input type="checkbox"/> Request for Corrected Filing Receipt with Enclosures <input type="checkbox"/> A self-addressed prepaid postcard for acknowledging receipt <input type="checkbox"/> Other Enclosure(s) <i>(please identify below):</i>
Remarks	<input checked="" type="checkbox"/> The Commissioner is hereby authorized to charge any additional fees required or credit any overpayments to Deposit Account No. 19-2380 for the above identified docket number.	

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT	
Firm or Individual name	Carlos R. Villamar Registration No. 43,224 Nixon Peabody LLP 401 9 th Street, N/W, Suite 900 Washington, D.C. 20004-2128
Signature	
Date	July 17, 2006

CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]	
I hereby certify that this correspondence is being:	
<input type="checkbox"/> deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop _____, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450	
<input type="checkbox"/> transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) _____.	
_____ Date	_____ Signature
	_____ Typed or printed name



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:)	Confirmation No.: 8299
Mai NGUYEN, <i>et al.</i>)	Group Art Unit: 3621
Serial No. 10/956,070)	Examiner: Augustin Evens
Filed: October 4, 2004)	
For: SYSTEM AND METHOD FOR RIGHTS OFFERING AND GRANTING USING SHARED STATE VARIABLES)	

U.S. Patent and Trademark Office
Customer Services Window, **Mail Stop AF**
Randolph Building
401 Dulany Street
Alexandria, VA 22314

AMENDMENT AFTER FINAL REJECTION

Sir:

In response to the final Office Action mailed **April 17, 2006**, the below amendments and following remarks are submitted in connection with the above-identified application.

Amendments to the Claims:

1. (Cancelled)

2. (Previously presented) The method of claim 40, wherein the state variable in the first or second license inherits a state thereof for content usage or rights derivation from other generated usage rights and meta-rights.

3. (Previously presented) The method of claim 40, wherein the state variable in the first or second license shares a state thereof for content usage or rights derivation with other generated usage rights and meta-rights.

4. (Previously presented) The method of claim 40, wherein the state variable in the first or second license inherits a remaining state for content usage or rights derivation from other generated usage rights and meta-rights.

5. (Previously presented) The method of claim 40, wherein the state variable in the first or second license is updated upon exercise of a right associated with the state variable.

6. (Previously presented) The method of claim 40, wherein the state variable in the first or second license represents a collection of states.

7. (Currently amended) The method of claim 40, further comprising:
generating in a third license one or more rights from at least one of the usage right ~~and/or~~ and the meta-right in the second license,

wherein the one or more rights in the third license includes at least one right that is shared among one or more users or devices;

associating at least one state variable with the at least one right that is shared in the third license,

wherein the at least one state variable that is associated with the third license is based on the at least one state variable that is associated with the second license.

8. (Previously presented) The method of claim 40, further comprising a plurality of state variables that determine the state of the at least one right that is shared in the first or the second license.

9. (Cancelled)

10. (Currently amended) The method of claim 40, wherein the state variable in the second license is transferred from the ~~usage rights and meta-rights~~ at least one right in the first license and is associated with the right that is shared in the second license.

11-13. (Cancelled)

14. (Previously presented) The system of claim 41, wherein the state variable in the first or second license inherits a state thereof for content usage or rights derivation from other generated usage rights and meta-rights.

15. (Previously presented) The system of claim 41, wherein the state variable in the first or second license shares a state thereof for content usage or rights derivation with other generated usage rights and meta-rights.

16. (Previously presented) The system of claim 41, wherein the state variable in the first or second license inherits a remaining state for content usage or rights derivation from other generated usage rights and meta-rights.

17. (Previously presented) The system of claim 41, wherein the state variable in the first or second license is updated upon exercise of a right associated with the state variable.

18. (Previously presented) The system of claim 41, wherein the state variable in the first or second license represents a collection of states.

19. (Currently amended) The system of claim 41, further comprising:
means for generating in a third license one or more rights from at least one of the usage right ~~and/or~~ and the meta-right in the second license,

wherein the one or more rights in the third license includes at least one right that is shared among one or more users or devices;

means for associating at least one state variable with the at least one right that is shared in the third license,

wherein the at least one state variable that is associated with the third license is based on the at least one state variable that is associated with the second license.

20. (Previously presented) The system of claim 41, including a plurality of state variables that determine the state of the at least one right that is shared in the first or the second license.

21. (Cancelled)

22. (Currently amended) The system of claim 41, wherein the state variable in the second license is transferred from the ~~usage rights and meta rights~~ at least one right in the first license and is associated with the right that is shared in the second license.

23-24. (Cancelled)

25. (Currently amended) The system of claim 41, wherein the system is implemented with one or more hardware ~~and/or~~ and software components.

26. (Cancelled)

27. (Previously presented) The device of claim 42, wherein the state variable in the first or second license inherits a state thereof for content usage or rights derivation from other generated usage rights and meta-rights.

28. (Previously presented) The device of claim 42, wherein the state variable in the first or second license shares a state thereof for content usage or rights derivation with other generated usage rights and meta-rights.

29. (Previously presented) The device of claim 42, wherein the state variable in the first or second license inherits a remaining state for content usage or rights derivation from other generated usage rights and meta-rights.

30. (Previously presented) The device of claim 42, wherein the state variable in the first or second license is updated upon exercise of a right associated with the state variable.

31. (Previously presented) The device of claim 42, wherein the state variable in the first or second license represents a collection of states.

32. (Currently amended) The device of claim 42, wherein a third license includes one or more rights from at least one of the usage right ~~and/or~~ and the meta-right in the second license,

the one or more rights in the third license includes at least one right that is shared among one or more users or devices,

at least one state variable is associated with the at least one right that is shared in the third license, and

the at least one state variable that is associated with the third license is based on the at least one state variable that is associated with the second license.

33. (Previously presented) The device of claim 42, including a plurality of state variables that determine the state of the at least one right that is shared in the first or the second license.

34. (Cancelled)

35. (Currently amended) The device of claim 42, wherein the state variable in the second license is transferred from the ~~usage rights and meta-rights~~ at least one right in the first license and is associated with the right that is shared in the second license.

36-39. (Cancelled)

40. (Currently amended) A method for sharing rights adapted to be associated with an item, the method comprising:

specifying in a first license at least one usage right ~~and/or~~ and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices;

defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item;

defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, wherein said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights;

~~associating at least one state variable with the at least one right in the first license and that, wherein the at least one state variable is shared among the one or more users or devices, wherein the at least one state variable and is used to determine how the shared right is further generated in a second license specified as unspecified or is assigned an identification in said first license, and said identification references a location where a state of rights is tracked;~~

generating in ~~[[the]]~~ a second license one or more rights ~~from the usage right and/or based on~~ the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices; and

associating at least one state variable with the at least one right that is shared in the second license, wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

41. (Currently amended) A system for sharing rights adapted to be associated with an item, the system comprising:

means for specifying in a first license at least one usage right ~~and/or~~ and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices;

means for defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item;

means for defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, wherein said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights;

means for associating at least one state variable with the at least one right in the first license ~~and that, wherein the at least one state variable~~ is shared among the one or more users or devices, ~~wherein the at least one state variable and is used to determine how the shared right is further generated in a second license~~ specified as unspecified or is assigned an identification in said first license, and said identification references a location where a state of rights is tracked;

means for generating in [[the]] a second license one or more rights ~~from the usage right and/or based on~~ the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices; and

means for associating at least one state variable with the at least one right that is shared in the second license, wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

42. (Currently amended) A device for sharing rights adapted to be associated with an item, the device comprising:

means for receiving a first license specifying at least one usage right ~~and/or and~~ and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices, the least one usage right defines a manner of use selected from a plurality of permitted manners of use for the item, the at least one meta-right defines a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights, at least one state variable is associated with the at least one right in the first license and is shared among the one or more users or devices and is specified as unspecified or is assigned an identification in said first license, and said identification references a location where a state of rights is tracked, ~~the at least one state variable is used to determine how the shared right is further generated in a second license;~~ and

means for generating in [[the]] a second license one or more rights ~~from the usage right and/or based on~~ the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices, at least one state variable is associated with the at least one right that is shared in the second

license, and the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

43. (Currently amended) The method of claim 40, wherein the method is implemented with one or more hardware ~~and/or~~ and software components configured to perform the steps of the method.

44. (Previously presented) The method of claim 40, wherein the method is implemented with one or more computer readable instructions embedded on a computer readable medium and configured to cause one or more computer processors to perform the steps of the method.

45. (Currently amended) The device of claim 42, wherein the device is implemented with one or more hardware ~~and/or~~ and software components.

46-48. (Cancelled)

49. (Currently amended) The method of claim 40, wherein the plurality of permitted manners of use for the item include copy, transfer, loan, play, print, ~~back-up, restore,~~ delete, extract, embed, edit, authorize, install, and un-install the item.

50. (Currently amended) The system of claim 41, wherein the plurality of permitted manners of use for the item include copy, transfer, loan, play, print, ~~back-up, restore,~~ delete, extract, embed, edit, authorize, install, and un-install the item.

51. (Currently amended) The device of claim 42, wherein the plurality of permitted manners of use for the item include copy, transfer, loan, play, print, ~~back-up, restore,~~ delete, extract, embed, edit, authorize, install, and un-install the item.

52. (Previously presented) The method of claim 40, wherein the plurality of permitted manners of rights derivation for the item include issue, modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, and revoke rights for the item.

53. (Previously presented) The system of claim 41, wherein the plurality of permitted manners of rights derivation for the item include issue, modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, and revoke rights for the item.

54. (Previously presented) The device of claim 42, wherein the plurality of permitted manners of rights derivation for the item include issue, modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, and revoke rights for the item.

REMARKS

The following amendments and remarks are submitted to be fully responsive to the final Official Action of **April 17, 2006**. In the present response, claims 7, 10, 19, 22, 25, 31, 32, 40-43, 45 and 49-51 are amended, and claims 9, 21, 34, and 46-48 are cancelled without prejudice or disclaimer. No new matter is introduced (see, e.g., Applicants' published Specification ¶¶ [0007], [0009], [0093] and [0098]). Thus, claims 2-8, 10, 14-20, 22, 25, 27-33, 35, 40-45, and 49-54 are still pending. Reconsideration and allowance of this application are respectfully requested.

Referring now to the present Office Action, claims 40-43, 45, 7, 19, 25, and 32 were rejected under 35 U.S.C. §112, second paragraph, based on indefiniteness. In response, claims 7, 19, 25, 32, 40, 41, 42, 43 and 45 are amended to correct the noted and discovered informalities. Claims 7, 19, and 32 and 49-53 were rejected under 35 U.S.C. §112, first paragraph, based on non-described subject matter. In response, claims 49-51 are amended to correct the noted and discovered informalities. With respect to the third license, as recited in claims 7, 19, and 32, this feature is clearly described in Applicants' Specification (see, e.g., Applicants' published Specification FIGs. 16 and 18 and the description thereof). With respect to exercise and extract, as recited in claims 52-54, this feature is clearly described in Applicants' Specification (see, e.g., Applicants' published Specification ¶¶ [0007] and [0009]). Accordingly, all of the claims are in compliance with 35 U.S.C. §112 and no further rejection on such basis is anticipated. If, however, the Examiner disagrees, the Examiner is invited to contact the undersigned attorney, who will work with the Examiner in a mutual effort to derive satisfactory claim language.

Claims 2-10, 14-22, 25, 27-35, and 40-54 also were rejected under 35 U.S.C. §102(b) as being anticipated by U.S. Patent No. 6,226,618 to *Downs et al.* However, claims 2-10, 14-22, 25, 27-35, and 40-54 are patentably distinguishable over *Downs et al.*, because *Downs et al.* fails to disclose, teach or suggest all of the features recited in the present claims, as amended. For example, independent claim 40 (emphasis added) recites:

A method for sharing rights adapted to be associated with an item, the method comprising:
specifying in a first license at least one usage right and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices;
defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item;
defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights

derivation for the item, wherein said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights;

associating at least one state variable with the at least one right in the first license, wherein the at least one state variable is shared among the one or more users or devices, and is specified as unspecified or is assigned an identification in said first license, and said identification references a location where a state of rights is tracked;

generating in a second license one or more rights based on the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices; and

associating at least one state variable with the at least one right that is shared in the second license, wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

Independent claim 41 (emphasis added) recites:

A system for sharing rights adapted to be associated with an item, the system comprising:

means for specifying in a first license at least one usage right and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices;

means for defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item;

means for defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, wherein said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights;

means for associating at least one state variable with the at least one right in the first license, wherein the at least one state variable is shared among the one or more users or devices, and is specified as unspecified or is assigned an identification in said first license, and said identification references a location where a state of rights is tracked;

means for generating in a second license one or more rights based on the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices; and

means for associating at least one state variable with the at least one right that is shared in the second license, wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

Independent claim 42 (emphasis added) recites:

A device for sharing rights adapted to be associated with an item, the device comprising:

means for receiving a first license specifying at least one usage right and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices, the least one usage right defines a manner of use selected from a plurality of permitted manners of use for the item, the at least one meta-right defines a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights, at

least one state variable is associated with the at least one right in the first license and is shared among the one or more users or devices and is specified as unspecified or is assigned an identification in said first license, and said identification references a location where a state of rights is tracked; and

means for generating in a second license one or more rights based on the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices, at least one state variable is associated with the at least one right that is shared in the second license, and the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

Thus, the present invention recited in independent claims 40, 41 and 42 includes the novel features of specifying in a first license at least one usage right and at least one meta-right for an item, the at least one meta-right allows one or more users or devices to transfer rights or to derive new rights, associating at least one state variable with the at least one right in the first license, wherein the at least one state variable is shared among the one or more users or devices, and is specified as unspecified or is assigned an identification in the first license, and the identification references a location where a state of rights is tracked, generating in a second license one or more rights based on the meta-right in the first license, and associating at least one state variable with at least one right that is shared in the second license, wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

By contrast, *Downs et al.* is directed to a method and apparatus of securely providing data to a user's system, wherein the data is encrypted so as to only be decryptable by a data decrypting key, the data decrypting key being encrypted using a first public key, and the encrypted data being accessible to the user's system. The method includes transferring the encrypted data decrypting key to a clearing house that possesses a first private key, which corresponds to the first public key; decrypting the data decrypting key using the first private key; re-encrypting the data decrypting key using a second public key; transferring the re-encrypted data decrypting key to the user's system, the user's system possessing a second private key, which corresponds to the second public key; and decrypting the re-encrypted data decrypting key using the second private key. However, *Downs et al.* fails to disclose, teach or suggest at least the noted features recited in independent claims 40, 41 and 42.

Accordingly, the portions of *Downs et al.* cited in the present Office Action do support a further rejection. For example, in column 21, lines 30-33, *Downs et al.* states that:

... the Content Provider sets the allowable Usage Conditions and transmits them to the Electronic Digital Content Store in a SC. The Electronic Digital Content Store can add to or narrow the Usage Conditions as long as it doesn't invalidate the original conditions set by the Content Provider.

According to the above passage, however, what is transferred to the content store is a set of allowable usage conditions that the store can modify before sending out to an end-user device. Examples of usage conditions include "Song is recordable," "Song can be played n number of times," etc. Usage conditions are merely conditions for the end user and do not include rights given to the content store. Thus, the content store merely can modify and transfer the allowable usage conditions, but otherwise does not have any rights to transfer rights or to derive new rights, as recited in independent claims 40, 41 and 42.

This is further evidenced in column 26, lines 61-67, wherein *Downs et al.* describes the concept of using a template to indicate which information can be modified, as follows:

An Offer SC(s) 641 template in the Metadata SC(s) 620 indicates which information can be overridden by the Electronic Digital Content Store(s) 103 in the Offer SC(s) 641 and what, if any, additional information is required by the Electronic Digital Content Store(s) 103 and what parts are retained in the embedded Metadata SC(s) 620.

Accordingly, in *Downs et al.*, both the right to transfer usage conditions and the right to modify rights stop at the content store. On a user device, a user may be permitted to make copies of content. But, since the user does not have the right to issue new rights, a content copy inherits the exact same usage rights as the original copy, with the copy count updated to reflect the number of copies made, as noted in column 21, lines 58-60 of *Downs et al.*:

The End-User Device(s) 109 also appropriately updates the copy/play code in the original copy of the Content 113 and on any new secondary copy.

The disadvantage of the copy conditions described by *Downs et al.* is that they are copied but not transferred. Specifically, DRM systems, such as that of *Downs et al.*, employ a key that can be used to decrypt protected content in a license having a copy condition, wherein the key is bound to a specific device in such a way that only the specific device can use the key to decrypt the content. Accordingly, a copy of a license does not allow a second device to use the content. Therefore, the copy condition described by *Downs et al.* does not disclose, teach, or suggest the transfer of rights nor meta-rights.

Accordingly, the invention recited in independent claims 40, 41 and 42 includes the recognition that (see, e.g., Applicants' published Specification ¶ [0009]):

... there are limitations associated with the above-mentioned paradigms wherein only usage rights and conditions associated with content are specified by content owners or other grantors of rights. Once purchased by an end user, a consumer, or a distributor, of content along with its associated usage rights and conditions has no means to be legally passed on to a next recipient in a distribution chain.

Advantageously, the invention recited in independent claims 40, 41 and 42 solves such problems by employing usage rights and meta-rights for an item in the manner claimed. For example, FIG. 18 of Applicants' published Specification teaches that a first license 1801 from a content provider allows any affiliated club to transfer a right to play an e-book to one of its members. The license 1801 specifies a meta-right (e.g., issue) and a usage right (e.g., play). When Acme joins the affiliation program, Acme gets a second license 1802 permitting it to transfer the right to read the e-book to one of its members. Alice is an Acme club member and requests to access the e-book and receives a third license 1804 granting her the right to read the e-book. In this example, Alice's license is different than Acme's license, and is different than the merely assigning usage conditions to a copy of content, as disclosed by *Downs et al.*

As noted above, a main difference between *Downs et al.* regarding meta-rights is that the invention recited in independent claims 40, 41 and 42 enables meta-rights to be passed down to multiple value chain participants, while the usage conditions in *Downs et al.* are limited. Accordingly, *Downs et al.* does not disclose, teach or suggest passing usage conditions beyond the content store (i.e., copy right is not a meta right) nor addresses a need to modify rights outside the content store.

With respect to state variables, *Downs et al.* does not disclose, teach or suggest employing a state variable in the manner recited in independent claims 40, 41 and 42, for example, to control sharing of content or to represent various other states. In this respect, the present Office Action cites column 59, line 50, of *Downs et al.*, which is directed to "the number of playable copies the End-User(s) is allowed to make." However, the cited portion does not disclose, teach or suggest at least one state variable that is associated with a second license is based on at least one state variable that is associated with a first license, as recited in independent claims 40, 41 and 42. For example, as described with respect to FIG. 16 of Applicants' published Specification, a content provider specifies in a first license 1601 that

up to 5 club members can play an e-book at the same time. Attached to such right is an unspecified state variable 1607, which represents a modifiable field. Before such right is transferred to the Acme club, the unspecified state variable 1607 is changed to reference “urn:acme:club.” Alice, a club member, requests and receives a third license 1604 allowing her to play the e-book. Bob, another club member also requests and receives a fourth license 1605 to play the e-book. When each of Alice and Bob plays the e-book, the same state variable 1608, which has been created in the rights enforcing device, is updated. As described with respect to FIG. 18 of Applicants’ published Specification, a state variable 1808 also can remain unspecified in a transferred license 1802 and can be modified by a consumer thereof.

Downs et al. is similarly deficient at column 59, lines 55-60, which merely discloses a “period of time during which the purchase/rental transaction is allowed to occur,” which is simply a validity interval condition that specifies an exact time interval, and is no different than the count condition described in column 59, line 50 of *Downs et al.* In addition, although the present Office Action states that a count of 5 is a state variable, with the invention recited in independent claims 40, 41 and 42 a state variable identification can be attached to a counter, wherein the state variable identification is the identification of a state variable which keeps track of, for example, a count of 5.

The remaining portions of *Downs et al.* are similarly deficient. Accordingly, *Downs et al.* fails to disclose, teach or suggest the novel features of specifying in a first license at least one usage right and at least one meta-right for an item, the at least one meta-right allows one or more users or devices to transfer rights or to derive new rights, associating at least one state variable with the at least one right in the first license, wherein the at least one state variable is shared among the one or more users or devices, and is specified as unspecified or is assigned an identification in the first license, and the identification references a location where a state of rights is tracked, generating in a second license one or more rights based on the meta-right in the first license, and associating at least one state variable with at least one right that is shared in the second license, wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license, as recited in independent claims 40, 41 and 42.

As noted above, the invention recited in independent claims 40, 41 and 42 and claims dependent therefrom recognizes and solves the following problems:

[0009] However, there are limitations associated with the above-mentioned paradigms wherein only usage rights and conditions associated with content are specified by content owners or other grantors of rights. Once purchased by an end user, a consumer, or a distributor, of content along with its associated usage rights and conditions has no means to be legally passed on to a next recipient in a distribution chain. Further the associated usage rights have no provision for specifying rights to derive other rights, i.e. Rights to modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, revoke, or the like. Common content distribution models often include a multi-tier distribution and usage chain. Known DRM systems do not facilitate the ability to prescribe rights and conditions for all participants along a content distribution and usage chain. Therefore, it is difficult for a content owner to commercially exploit content unless the owner has a relationship with each party in the distribution chain.

The invention recited in independent claims 40, 41 and 42 and claims dependent therefrom provides the following advantages:

[0090] There are multiple ways to specify the scope of state variables, each of which can affect whether the derivative state variables can be shared, how the derivative state variables can be shared, and the like. For example, a state variable can be local, and solely confined to a recipient or can be global, and shared by a predetermined group of recipients. A global state variable can be shared by a group of recipients not determined when derived rights are issued, but to be specified later, perhaps based on certain rules defined in the license or based on other means. A global state variable can be shared between one or more rights suppliers, predetermined recipients, un-specified recipients, and the like. Advantageously, depending on the sharing employed with a given a business model and the rights granted in the meta-rights, state variables can be created at different stages of the value chain.

By contrast, *Downs et al.* fails disclose, teach or suggest the noted features, fails to recognize or solve the noted problems, and fails to provide the advantages of the invention recited in independent claims 40, 41 and 42. The dependent claims are allowable on their own merits and for at least the reasons as argued above with respect to independent claims 40, 41 and 42.

The present amendment is submitted in accordance with the provisions of 37 C.F.R. §1.116, which after Final Rejection permits entry of amendments placing the claims in better form for consideration on appeal. As the present amendment is believed to overcome outstanding rejections under 35 U.S.C. §102, the present amendment places the application in better form for consideration on. It is therefore respectfully requested that 37 C.F.R. §1.116 be liberally construed, and that the present amendment be entered.

In view of the foregoing, it is submitted that the present application is in condition for allowance and a notice to that effect is respectfully requested. However, if the Examiner deems that any issue remains after considering this response, the Examiner is invited to contact the undersigned attorney to expedite the prosecution and engage in a joint effort to work out a mutually satisfactory solution.

Respectfully submitted,

NIXON PEABODY, LLP

/Carlos R. Villamar, Reg. # 43,224/
Carlos R. Villamar
Reg. No. 43,224

NIXON PEABODY LLP
CUSTOMER NO.: 22204
401 9th Street, N.W., Suite 900
Washington, DC 20004
Tel: 202-585-8000
Fax: 202-585-8080

PATENT APPLICATION FEE DETERMINATION RECORD
Effective October 1, 2004

Application or Docket Number

10956070

CLAIMS AS FILED - PART I

	(Column 1)	(Column 2)
TOTAL CLAIMS	39	
FOR	NUMBER FILED	NUMBER EXTRA
TOTAL CHARGEABLE CLAIMS	39 minus 20 =	19
INDEPENDENT CLAIMS	3 minus 3 =	0
MULTIPLE DEPENDENT CLAIM PRESENT <input type="checkbox"/>		

* If the difference in column 1 is less than zero, enter "0" in column 2

SMALL ENTITY TYPE OR

OTHER THAN SMALL ENTITY

RATE	FEE		RATE	FEE
BASIC FEE	395.00	OR	BASIC FEE	790.00
X\$ 9=		OR	X\$18=	342.00
X44=		OR	X88=	
+150=		OR	+300=	
TOTAL		OR	TOTAL	1132.00

CLAIMS AS AMENDED - PART II

24/06

	(Column 1)	(Column 2)	(Column 3)
AMENDMENT A	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
Total	43	Minus ** 39	= 4
Independent	3	Minus *** 3	= 0
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>			

SMALL ENTITY OR

OTHER THAN SMALL ENTITY

RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
X\$ 9=		OR	X\$18=	200.00
X44=		OR	X88=	
+150=		OR	+300=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	200.00

7/17/06

	(Column 1)	(Column 2)	(Column 3)
AMENDMENT B	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
Total	37	Minus ** 39	= 0
Independent	3	Minus *** 3	= 0
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>			

SMALL ENTITY OR

OTHER THAN SMALL ENTITY

RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
X\$ 9=		OR	X\$18=	
X44=		OR	X88=	
+150=		OR	+300=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

	(Column 1)	(Column 2)	(Column 3)
AMENDMENT C	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
Total		Minus **	=
Independent		Minus ***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>			

SMALL ENTITY OR

OTHER THAN SMALL ENTITY

RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
X\$ 9=		OR	X\$18=	
X44=		OR	X88=	
+150=		OR	+300=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
 ** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
 *** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
 The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
-----------------	-------------	----------------------	---------------------	------------------

10/956,070	10/04/2004	Mai Nguyen	111325-235000	8299
------------	------------	------------	---------------	------

22204	7590	08/11/2006
-------	------	------------

NIXON PEABODY, LLP
401 9TH STREET, NW
SUITE 900
WASHINGTON, DC 20004-2128

EXAMINER

AUGUSTIN, EVENS J

ART UNIT	PAPER NUMBER
----------	--------------

3621

DATE MAILED: 08/11/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Advisory Action Before the Filing of an Appeal Brief	Application No. 10/956,070	Applicant(s) NGUYEN ET AL.	
	Examiner Evens Augustin	Art Unit 3621	

--The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

THE REPLY FILED 17 July 2006 FAILS TO PLACE THIS APPLICATION IN CONDITION FOR ALLOWANCE.

1. The reply was filed after a final rejection, but prior to or on the same day as filing a Notice of Appeal. To avoid abandonment of this application, applicant must timely file one of the following replies: (1) an amendment, affidavit, or other evidence, which places the application in condition for allowance; (2) a Notice of Appeal (with appeal fee) in compliance with 37 CFR 41.31; or (3) a Request for Continued Examination (RCE) in compliance with 37 CFR 1.114. The reply must be filed within one of the following time periods:
- a) The period for reply expires 3 months from the mailing date of the final rejection.
- b) The period for reply expires on: (1) the mailing date of this Advisory Action, or (2) the date set forth in the final rejection, whichever is later. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of the final rejection.
- Examiner Note: If box 1 is checked, check either box (a) or (b). ONLY CHECK BOX (b) WHEN THE FIRST REPLY WAS FILED WITHIN TWO MONTHS OF THE FINAL REJECTION. See MPEP 706.07(f).

Extensions of time may be obtained under 37 CFR 1.136(a). The date on which the petition under 37 CFR 1.136(a) and the appropriate extension fee have been filed is the date for purposes of determining the period of extension and the corresponding amount of the fee. The appropriate extension fee under 37 CFR 1.17(a) is calculated from: (1) the expiration date of the shortened statutory period for reply originally set in the final Office action; or (2) as set forth in (b) above, if checked. Any reply received by the Office later than three months after the mailing date of the final rejection, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

NOTICE OF APPEAL

2. The Notice of Appeal was filed on _____. A brief in compliance with 37 CFR 41.37 must be filed within two months of the date of filing the Notice of Appeal (37 CFR 41.37(a)), or any extension thereof (37 CFR 41.37(e)), to avoid dismissal of the appeal. Since a Notice of Appeal has been filed, any reply must be filed within the time period set forth in 37 CFR 41.37(a).

AMENDMENTS

3. The proposed amendment(s) filed after a final rejection, but prior to the date of filing a brief, will not be entered because
- (a) They raise new issues that would require further consideration and/or search (see NOTE below);
- (b) They raise the issue of new matter (see NOTE below);
- (c) They are not deemed to place the application in better form for appeal by materially reducing or simplifying the issues for appeal; and/or
- (d) They present additional claims without canceling a corresponding number of finally rejected claims.

NOTE: _____. (See 37 CFR 1.116 and 41.33(a)).

4. The amendments are not in compliance with 37 CFR 1.121. See attached Notice of Non-Compliant Amendment (PTOL-324).
5. Applicant's reply has overcome the following rejection(s): _____.
6. Newly proposed or amended claim(s) _____ would be allowable if submitted in a separate, timely filed amendment canceling the non-allowable claim(s).
7. For purposes of appeal, the proposed amendment(s): a) will not be entered, or b) will be entered and an explanation of how the new or amended claims would be rejected is provided below or appended.
- The status of the claim(s) is (or will be) as follows:
- Claim(s) allowed: _____.
- Claim(s) objected to: _____.
- Claim(s) rejected: _____.
- Claim(s) withdrawn from consideration: _____.

AFFIDAVIT OR OTHER EVIDENCE

8. The affidavit or other evidence filed after a final action, but before or on the date of filing a Notice of Appeal will not be entered because applicant failed to provide a showing of good and sufficient reasons why the affidavit or other evidence is necessary and was not earlier presented. See 37 CFR 1.116(e).
9. The affidavit or other evidence filed after the date of filing a Notice of Appeal, but prior to the date of filing a brief, will not be entered because the affidavit or other evidence failed to overcome all rejections under appeal and/or appellant fails to provide a showing a good and sufficient reasons why it is necessary and was not earlier presented. See 37 CFR 41.33(d)(1).
10. The affidavit or other evidence is entered. An explanation of the status of the claims after entry is below or attached.

REQUEST FOR RECONSIDERATION/OTHER

11. The request for reconsideration has been considered but does NOT place the application in condition for allowance because: _____
12. Note the attached Information Disclosure Statement(s). (PTO/SB/08 or PTO-1449) Paper No(s). _____
13. Other: See Continuation Sheet.

[Handwritten Signature]
CALVIN L. HEWITTI
PRIMARY EXAMINER

Continuation of 13. Other: The newly added limitation of "wherein said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights" and "specified as unspecified or is assigned an identification in said first license, and said identification references a location where a state of rights is tracked" in independent claims 40-42 change the scope of the claims and will require further search and/or consideration.

**REQUEST
FOR
CONTINUED EXAMINATION (RCE)
TRANSMITTAL**

Subsection (b) of 35 U.S.C. § 132, effective on May 29, 2000,
provides for continued examination of an utility or plant application
filed on or after June 8, 1995.
See The American Inventors Protection Act of 1999 (AIPA).

Application Number	10/956,070
Filing Date	October 4, 2004
First Named Inventor	Mai NGUYEN, <i>et al.</i>
Group Art Unit	3621
Examiner Name	Evens J. Augustin
Attorney Docket Number	111325-235000

This is a Request for Continued Examination (RCE) under 37 C.F.R. § 1.114 of the above-identified application.
NOTE: 37 C.F.R. § 1.114 is effective on May 29, 2000. If the above-identified application was filed prior to May 29, 2000, applicant may wish to consider filing a continued prosecution application (CPA) under 37 C.F.R. § 1.53(d) (PTO/SB/29) instead of a RCE to be eligible for the patent term adjustment provisions of the AIPA. See Changes to Application Examination and Provisional Application Practice, Final Rule, 65 Fed. Reg. 50092 (Aug. 16, 2000); Interim Rule, 65 Fed. Reg. 14865 (Mar. 20, 2000), 1233 Off. Gaz. Pat. Office 47 (Apr. 11, 2000), which established RCE practice.

1. **Submission required under 37 C.F.R. § 1.114**
- a. Previously submitted
- i. Consider the amendment(s)/reply under 37 C.F.R. § 1.116 previously filed on July 17, 2006
(Any unentered amendment(s) referred to above will be entered).
- ii. Consider the arguments in the Appeal Brief or Reply Brief previously filed on _____
- iii. Other _____
- b. Enclosed
- i. Amendment/Reply
- ii. Affidavit(s)/Declaration(s)
- iii. Information Disclosure Statement (IDS)
- iv. Submission of Formal Drawings
- v. Petition for two-month Extension of Time
2. **Miscellaneous**
- a. Suspension of action on the above-identified application is requested under 37 C.F.R. § 1.103(c) for a period of _____ months. (Period of suspension shall not exceed 3 months; Fee under 37 C.F.R. § 1.17(l) **required**)
- b. Other _____
3. **Fees** The RCE fee under 37 C.F.R. § 1.17(e) is required by 37 C.F.R. § 1.114 when the RCE is filed.
- a. The Director is hereby authorized to charge the following fees, additional fees which may be required, or credit any overpayments, to Deposit Account No. 19-2380
- i. RCE fee required under 37 C.F.R. § 1.17(e)
- ii. Extension of time fee (37 C.F.R. §§ 1.136 and 1.17)
- iii. Other _____
- b. Check in the amount of \$ _____ enclosed
- c. Payment by credit card (Form PTO-2038 enclosed)

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT REQUIRED

Name (<i>Print/Type</i>)	Carlos R. Villamar	Registration No. (<i>Attorney/Agent</i>)	43,224
Signature	<u>/Carlos R. Villamar, Reg. # 43,224/</u>	Date	September 14, 2006

**CERTIFICATE OF MAILING OR TRANSMISSION
[37 CFR 1.8(a)]**

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to: Mail Stop RCE, Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450, or being facsimile transmitted to the USPTO at _____, on _____.

Signature:	
Name:	

SEND TO: Mail Stop RCE
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Electronic Patent Application Fee Transmittal

Application Number:	10956070
Filing Date:	04-Oct-2004
Title of Invention:	System and method for rights offering and granting using shared state variables
First Named Inventor:	Mai Nguyen
Filer:	Carlos Rafael Villamar/Kisha Avery
Attorney Docket Number:	111325-235000

Filed as Large Entity

Utility Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Extension - 2 months with \$0 paid	1252	1	450	450

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
Miscellaneous:				
Request for continued examination	1801	1	790	790
Total in USD (\$)				1240

Electronic Acknowledgement Receipt

EFS ID:	1202088
Application Number:	10956070
Confirmation Number:	8299
Title of Invention:	System and method for rights offering and granting using shared state variables
First Named Inventor:	Mai Nguyen
Customer Number:	22204
Filer:	Carlos Rafael Villamar/Kisha Avery
Filer Authorized By:	Carlos Rafael Villamar
Attorney Docket Number:	111325-235000
Receipt Date:	14-SEP-2006
Filing Date:	04-OCT-2004
Time Stamp:	17:52:52
Application Type:	Utility
International Application Number:	

Payment information:

Submitted with Payment	yes
Payment was successfully received in RAM	\$ 1240
RAM confirmation Number	523
Deposit Account	192380
The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows: Charge any Additional Fees required under 37 C.F.R. Section 1.16 and 1.17	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)	Multi Part	Pages
1	Request for Continued Examination (RCE)	111325_235000_Request_for_Continued_Examination.pdf	33135	no	1
Warnings:					
Information:					
2	Fee Worksheet (PTO-875)	fee-info.pdf	8332	no	2
Warnings:					
Information:					
Total Files Size (in bytes):			41467		
<p>This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.</p> <p><u>New Applications Under 35 U.S.C. 111</u> If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.</p> <p><u>National Stage of an International Application under 35 U.S.C. 371</u> If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.</p>					



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/956,070	10/04/2004	Mai Nguyen	111325-235000	8299

22204 7590 11/30/2006

NIXON PEABODY, LLP
401 9TH STREET, NW
SUITE 900
WASHINGTON, DC 20004-2128

EXAMINER

AUGUSTIN, EVENS J

ART UNIT PAPER NUMBER

3621

DATE MAILED: 11/30/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary

Application No. 10/956,070	Applicant(s) NGUYEN ET AL.	
Examiner Evens Augustin	Art Unit 3621	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) Responsive to communication(s) filed on 09/14/2006.
- 2a) This action is **FINAL**.
- 2b) This action is non-final.
- 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) Claim(s) 2-10, 14-22, 25, 27-35 and 40-54 is/are pending in the application.
4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) Claim(s) _____ is/are allowed.
- 6) Claim(s) 2-10, 14-22, 25, 27-35 and 40-54 is/are rejected.
- 7) Claim(s) _____ is/are objected to.
- 8) Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) The specification is objected to by the Examiner.
- 10) The drawing(s) filed on _____ is/are: a) accepted or b) objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
a) All b) Some * c) None of:
1. Certified copies of the priority documents have been received.
2. Certified copies of the priority documents have been received in Application No. _____.
3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) Notice of References Cited (PTO-892)
- 2) Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____
- 4) Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____
- 5) Notice of Informal Patent Application (PTO-152)
- 6) Other: _____

Status of Claims

1. Claims 2-10, 14-22, 25, 27-35, and 40-54 have been examined.

After further consideration and/search, it has been found that the newly added limitation of "wherein said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights" and "specified as unspecified or is assigned an identification in said first license, and said identification references a location where a state of rights is tracked" in independent claims 40-42 have been implicitly/explicitly in the prior art Downs et al.

2. The claims, as interpreted in light of the specification, are regarding content distribution system, in which the content owners specify the rights of distributors/end-users vis-à-vis the content. The present invention categorizes the rights associated with the content into **usage rights** and **meta-rights**. Usage rights are defined as a manner of use for the items, and meta-rights are defined as a manner of rights transfer for the items (page 4, par. 10). According to the applicant's specification, **meta-rights** can include rights to offer, grant, obtain, transfer, delegate, track, surrender, exchange, and revoke usage rights to/from others (page 9, par. 41). A **usage right** can be the right to print content **three times**. Each time the usage right is exercised, the value of the state variable "number of prints" is incremented. In this example, when the value of the state variable is **three**, the condition is no longer satisfied and content cannot be printed (page 9, par. 43). The content owner initially grants the distributor a license/rights to

Art Unit: 3621

distribute/sell the content, and the distributor can modify the initial licensing/rights from the owner to offer a more customized version of the rights to the end users (pages 24-25, par. 79-80)

Claim Rejections - 35 USC § 112

3. Claims 40-41 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. The claims include the language of “*specified as unspecified or is assigned an identification in said first license, and said identification references a location where a state of rights is tracked*”. It is not clear to one skilled in the art what distinctly is the subject matter the applicant is trying to claim.

4. The language will be interpreted as **keeping track of the content's copy/play usage** (column 20, lines 43-50, column 12, lines 11-12). Inherently the system keeps track of those devices in which the copies of the content are being transferred to and from, in order to keep track of the content's copy/play usage.

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) The invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Art Unit: 3621

2. Claims 2-10, 14-22, 25, 27-35, and 40-54 are rejected under 35 U.S.C. 102(b) as being anticipated by Downs et al (U.S 6226618).

As per claims 2-10, 14-22, 25, 27-35, and 40-54, Downs et al. disclose an invention that broadly relates to the field of electronic commerce and more particularly to a system and related tools for the secure delivery and rights management of digital assets, such as print media, films, games, and music over global communications networks such as the Internet and the World Wide Web. The invention includes the **means and devices (hardware and software combination)** (columns 53, lines 65-67, column 54, lines 1-3) to accomplish the items below-
Claims 25, 38, 39, 41-45, 48, 51, 54. The invention comprising of the following:

- Owners setting initial usage rights/licensing (**first license**) for content to the distributors (column 21, lines 30-33) – *Claims 40-42*
- Example of usage rights include (column 59, lines 38-69 – *Claims 40, 49-54*):

Compressed Version	384 Kbps
Type of user	Private Consumer
Type of Transaction	Purchase or Rental
Number Of copies	1
Rental Terms	14 Days
Transfer on What Media	Mini Disc or Computer

Art Unit: 3621

- Owners setting initial usage rights/licensing (**first license**) for content to the distributors (column 21, lines 30-33). Those usage rights can be modified by the digital store (column 21, lines 33-39) to create **secondary licensing** or customized licensing (column 10, lines 15-18) to the end user – *Claims 40-42, 10, 22, 35*
- Content providers (entity that supplies the content), providing (equivalent to generating) usage conditions (equivalent to usage rights), The content providers also stipulate that the content stores or distributors can add or narrow the original usage rights (meta-rights or rights derived from usage rights) (column 21, lines 30-36) - *Claims 40-42*
- Usage rights being copy restrictions, which is manner in which the content can be used (column 9, 32-34, col. 26, lines 10-12). - *Claims 40-42*
- Content stores or distributors can add or narrow the original usage rights (sub-rights) (column 21, lines 30-36) - *Claims 40-42*
- The system also defines the manner in which the content can be used (**meta-rights**) such as onto what kinds of media the content can be transferred to (column 59, lines 52-54) – *Claims 40-42, 46-48*
- State variable such the number of copies a user is allowed to make (column 59, line 50 or rental terms (column 59, lines 55-60) - *Claims 40-42*
- State variable such the number of copies a user is allowed to make (column 59, line 50 or rental terms (column 59, lines 55-60). Specify the number of plays and local copies allowed for the Content, and whether or not the Content may be recorded to an external portable device (state variable). Keeping track of the content's copy/play usage and update the copy/play status (column 20, lines 43-50, column 12, lines 11-12). Inherently

Art Unit: 3621

the system keeps track of those devices in which the copies of the content are being transferred to and from, in order to keep track of the content's copy/play usage - *Claims 40-42*

- The secondary licensing such as restrictions on rental time period can not violate the initial time period set by the initial licensing (column 21, line 35) - *Claims 40-42*
- The state variable is derived from the usage rights (column 59, line 50) or rental terms (column 59, lines 55-60) - *Claims 2-4, 14-16, 27-29*
- The system keeps track of the content's copy/play usage and updates the copy/play status (column 20, lines 49-50) - *Claims 5, 17, 30*
- A state variable can represent various other states, for example an item that rented can affect the number of copies that can be made or whether or not copies can be made - *Claims 6, 8, 18, 20, 31, 33*
- The system embeds a code on every copy the content, as it is transferred from user device to the next. When the Digital Content is accessed in a compliant End-User Devices, the End-User Player Application reads the watermark to check the use restrictions and updates the watermark as required. If the requested use of the content does not comply with the usage conditions, e.g., the number of copies has been exhausted, the End-User Device(s) will not perform the request (column 7, lines 40-55) - *Claims 7, 19, 32*
- The content does not specify how the initial set of rights and variable are to modified, as long as it does not violate the initial licensing (column 21, line 35) - *Claims 9, 21, 34*

Art Unit: 3621

Conclusion

3. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Evens Augustin whose telephone number is 571-272-6860. The examiner can normally be reached on 10am - 6pm M-F.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Andrew Fischer can be reached on (571)272-6779.

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is 571-272-6584.

Evens J. Augustin
November 26, 2006
Art Unit 3621

 11/27/06

ANDREW J. FISCHER
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 3600

Index of Claims



Application/Control No.

10/956,070

Examiner

Evans Augustin

Applicant(s)/Patent under Reexamination

NGUYEN ET AL.

Art Unit

3621

√	Rejected
=	Allowed

-	(Through numeral) Cancelled
+	Restricted

N	Non-Elected
I	Interference

A	Appeal
O	Objected

Claim		Date									
Final	Original	11/25/06									
	1	-									
	2	√									
	3	√									
	4	√									
	5	√									
	6	√									
	7	√									
	8	√									
	9	√									
	10	√									
	11	-									
	12	-									
	13	-									
	14	√									
	15	√									
	16	√									
	17	√									
	18	√									
	19	√									
	20	√									
	21	√									
	22	√									
	23	-									
	24	-									
	25	√									
	26	-									
	27	√									
	28	√									
	29	√									
	30	√									
	31	√									
	32	√									
	33	√									
	34	√									
	35	√									
	36	-									
	37	-									
	38	-									
	39	-									
	40	√									
	41	√									
	42	√									
	43	√									
	44	√									
	45	√									
	46	√									
	47	√									
	48	√									
	49	√									
	50	√									

Claim		Date									
Final	Original	11/25/06									
	51	√									
	52	√									
	53	√									
	54	√									
	55										
	56										
	57										
	58										
	59										
	60										
	61										
	62										
	63										
	64										
	65										
	66										
	67										
	68										
	69										
	70										
	71										
	72										
	73										
	74										
	75										
	76										
	77										
	78										
	79										
	80										
	81										
	82										
	83										
	84										
	85										
	86										
	87										
	88										
	89										
	90										
	91										
	92										
	93										
	94										
	95										
	96										
	97										
	98										
	99										
	100										

Claim		Date									
Final	Original										
	101										
	102										
	103										
	104										
	105										
	106										
	107										
	108										
	109										
	110										
	111										
	112										
	113										
	114										
	115										
	116										
	117										
	118										
	119										
	120										
	121										
	122										
	123										
	124										
	125										
	126										
	127										
	128										
	129										
	130										
	131										
	132										
	133										
	134										
	135										
	136										
	137										
	138										
	139										
	140										
	141										
	142										
	143										
	144										
	145										
	146										
	147										
	148										
	149										
	150										



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/956,070	10/04/2004	Mai Nguyen	111325-235000	8299

22204 7590 11/30/2006

NIXON PEABODY, LLP
401 9TH STREET, NW
SUITE 900
WASHINGTON, DC 20004-2128

EXAMINER

AUGUSTIN, EVENS J

ART UNIT PAPER NUMBER

3621

DATE MAILED: 11/30/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 10/956,070	Applicant(s) NGUYEN ET AL.	
	Examiner Evens Augustin	Art Unit 3621	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) Responsive to communication(s) filed on 09/14/2006.
- 2a) This action is **FINAL**.
- 2b) This action is non-final.
- 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) Claim(s) 2-10, 14-22, 25, 27-35 and 40-54 is/are pending in the application.
 - 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) Claim(s) _____ is/are allowed.
- 6) Claim(s) 2-10, 14-22, 25, 27-35 and 40-54 is/are rejected.
- 7) Claim(s) _____ is/are objected to.
- 8) Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) The specification is objected to by the Examiner.
- 10) The drawing(s) filed on _____ is/are: a) accepted or b) objected to by the Examiner.
 Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
 Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
 - a) All b) Some * c) None of:
 - 1. Certified copies of the priority documents have been received.
 - 2. Certified copies of the priority documents have been received in Application No. _____.
 - 3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) Notice of References Cited (PTO-892)
- 2) Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
 Paper No(s)/Mail Date _____.
- 4) Interview Summary (PTO-413)
 Paper No(s)/Mail Date _____.
- 5) Notice of Informal Patent Application (PTO-152)
- 6) Other: _____.

Status of Claims

1. Claims 2-10, 14-22, 25, 27-35, and 40-54 have been examined.

The claims, as interpreted in light of the specification, are regarding content distribution system, in which the content owners specify the rights of distributors/end-users vis-à-vis the content. The present invention categorizes the rights associated with the content into **usage rights** and **meta-rights**. Usage rights are defined as a manner of use for the items, and meta-rights are defined as a manner of rights transfer for the items (page 4, par. 10). According to the applicant's specification, **meta-rights** can include rights to offer, grant, obtain, transfer, delegate, track, surrender, exchange, and revoke usage rights to/from others (page 9, par. 41). A **usage right** can be the right to print content **three times**. Each time the usage right is exercised, the value of the state variable "number of prints" is incremented. In this example, when the value of the state variable is **three**, the condition is no longer satisfied and content cannot be printed (page 9, par. 43). The content owner initially grants the distributor a license/rights to distribute/sell the content, and the distributor can modify the initial licensing/rights from the owner to offer a more customized version of the rights to the end users (pages 24-25, par. 79-80)

After further consideration and/search, it has been found that the newly added limitation of "wherein said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights" and "specified as unspecified or is assigned an identification in said first license, and said identification references a location where a state of rights is tracked" in independent claims 40-42 have been implicitly/explicitly in the prior art Downs et al.

Claim Rejections - 35 USC § 112

1. Claims 40-41 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. The claims include the language of “*specified as unspecified or is assigned an identification in said first license, and said identification references a location where a state of rights is tracked*”. It is not clear to one skilled in the art what distinctly is the subject matter the applicant is trying to claim.
2. The language will be interpreted as **keeping track of the content's copy/play usage** (column 20, lines 43-50, column 12, lines 11-12). Inherently the system keeps track of those devices in which the copies of the content are being transferred to and from, in order to keep track of the content's copy/play usage.

Claim Rejections - 35 USC § 102

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) The invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.
3. Claims 2-10, 14-22, 25, 27-35, and 40-54 are rejected under 35 U.S.C. 102(b) as being anticipated by Downs et al (U.S 6226618).

Art Unit: 3621

As per claims 2-10, 14-22, 25, 27-35, and 40-54, Downs et al. disclose an invention that broadly relates to the field of electronic commerce and more particularly to a system and related tools for the secure delivery and rights management of digital assets, such as print media, films, games, and music over global communications networks such as the Internet and the World Wide Web. The invention includes the **means and devices (hardware and software combination)** (columns 53, lines 65-67, column 54, lines 1-3) to accomplish the items below-
Claims 25, 38, 39, 41-45, 48, 51, 54. The invention comprising of the following:

- Owners setting initial usage rights/licensing (**first license**) for content to the distributors (column 21, lines 30-33) – *Claims 40-42*
- Example of usage rights include (column 59, lines 38-69 – *Claims 40, 49-54*:

Compressed Version	384 Kbps
Type of user	Private Consumer
Type of Transaction	Purchase or Rental
Number Of copies	1
Rental Terms	14 Days
Transfer on What Media	Mini Disc or Computer

- Owners setting initial usage rights/licensing (**first license**) for content to the distributors (column 21, lines 30-33). Those usage rights can be modified by the digital store (column 21, lines 33-39) to create **secondary licensing** or customized licensing (column 10, lines 15-18) to the end user – *Claims 40-42, 10, 22, 35*

Art Unit: 3621

- Content providers (entity that supplies the content), providing (equivalent to generating) usage conditions (equivalent to usage rights), The content providers also stipulate that the content stores or distributors can add or narrow the original usage rights (meta-rights or rights derived from usage rights) (column 21, lines 30-36) - *Claims 40-42*
- Usage rights being copy restrictions, which is manner in which the content can be used (column 9, 32-34, col. 26, lines 10-12). - *Claims 40-42*
- Content stores or distributors can add or narrow the original usage rights (sub-rights) (column 21, lines 30-36) - *Claims 40-42*
- The system also defines the manner in which the content can be used (**meta-rights**) such as onto what kinds of media the content can be transferred to (column 59, lines 52-54) – *Claims 40-42, 46-48*
- State variable such the number of copies a user is allowed to make (column 59, line 50 or rental terms (column 59, lines 55-60) - *Claims 40-42*
- State variable such the number of copies a user is allowed to make (column 59, line 50 or rental terms (column 59, lines 55-60). Specify the number of plays and local copies allowed for the Content, and whether or not the Content may be recorded to an external portable device (state variable). Keeping track of the content's copy/play usage and update the copy/play status (column 20, lines 43-50, column 12, lines 11-12). Inherently the system keeps track of those devices in which the copies of the content are being transferred to and from, in order to keep track of the content's copy/play usage - *Claims 40-42*

Art Unit: 3621

- The secondary licensing such as restrictions on rental time period can not violate the initial time period set by the initial licensing (column 21, line 35) - *Claims 40-42*
- The state variable is derived from the usage rights (column 59, line 50) or rental terms (column 59, lines 55-60) - *Claims 2-4, 14-16, 27-29*
- The system keeps track of the content's copy/play usage and updates the copy/play status (column 20, lines 49-50) – *Claims 5, 17, 30*
- A state variable can represent various other states, for example an item that rented can affect the number of copies that can be made or whether or not copies can be made - *Claims 6, 8, 18, 20, 31, 33*
- The system embeds a code on every copy the content, as it is transferred from user device to the next. When the Digital Content is accessed in a compliant End-User Devices, the End-User Player Application reads the watermark to check the use restrictions and updates the watermark as required. If the requested use of the content does not comply with the usage conditions, e.g., the number of copies has been exhausted, the End-User Device(s) will not perform the request (column 7, lines 40-55) - *Claims 7, 19, 32*
- The content does not specify how the initial set of rights and variable are to modified, as long as it does not violate the initial licensing (column 21, line 35) - *Claims 9, 21, 34*


Art Unit: 3621


Conclusion

4. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Evens Augustin whose telephone number is 571-272-6860. The examiner can normally be reached on 10am - 6pm M-F.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Andrew Fischer can be reached on (571)272-6779.

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is 571-272-6584.


Evens J. Augustin
November 25, 2006
Art Unit 3621


ANDREW J. FISCHER
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 3600

Index of Claims



Application/Control No.

10/956,070

Examiner

Evens Augustin

Applicant(s)/Patent under Reexamination

NGUYEN ET AL.

Art Unit

3621

√	Rejected
=	Allowed

-	(Through numeral) Cancelled
+	Restricted

N	Non-Elected
I	Interference

A	Appeal
O	Objected

Claim		Date									
Final	Original	11/25/06									
	1	-									
	2	√									
	3	√									
	4	√									
	5	√									
	6	√									
	7	√									
	8	√									
	9	√									
	10	√									
	11	-									
	12	-									
	13	-									
	14	√									
	15	√									
	16	√									
	17	√									
	18	√									
	19	√									
	20	√									
	21	√									
	22	√									
	23	-									
	24	-									
	25	√									
	26	-									
	27	√									
	28	√									
	29	√									
	30	√									
	31	√									
	32	√									
	33	√									
	34	√									
	35	-									
	36	-									
	37	-									
	38	-									
	39	-									
	40	√									
	41	√									
	42	√									
	43	√									
	44	√									
	45	√									
	46	√									
	47	√									
	48	√									
	49	√									
	50	√									

Claim		Date									
Final	Original	11/25/06									
	51	√									
	52	√									
	53	√									
	54	√									
	55										
	56										
	57										
	58										
	59										
	60										
	61										
	62										
	63										
	64										
	65										
	66										
	67										
	68										
	69										
	70										
	71										
	72										
	73										
	74										
	75										
	76										
	77										
	78										
	79										
	80										
	81										
	82										
	83										
	84										
	85										
	86										
	87										
	88										
	89										
	90										
	91										
	92										
	93										
	94										
	95										
	96										
	97										
	98										
	99										
	100										

Claim		Date									
Final	Original										
	101										
	102										
	103										
	104										
	105										
	106										
	107										
	108										
	109										
	110										
	111										
	112										
	113										
	114										
	115										
	116										
	117										
	118										
	119										
	120										
	121										
	122										
	123										
	124										
	125										
	126										
	127										
	128										
	129										
	130										
	131										
	132										
	133										
	134										
	135										
	136										
	137										
	138										
	139										
	140										
	141										
	142										
	143										
	144										
	145										
	146										
	147										
	148										
	149										
	150										

REMARKS

The following amendments and remarks are submitted to be fully responsive to the non-final Official Action of **November 30, 2006**. In the present response, claims 40-42 are amended, and claims 55-57 are added. No new matter is introduced. Thus, claims 2-8, 10, 14-20, 22, 25, 27-33, 35, 40-45, and 49-57 are pending. Reconsideration and allowance of this application are respectfully requested.

Referring now to the present Office Action, claims 40-41 were rejected under 35 U.S.C. §112, second paragraph, based on indefiniteness. In response, claims 40-42 are amended to correct the noted and discovered informalities. Accordingly, all of the claims are in compliance with 35 U.S.C. §112 and no further rejection on such basis is anticipated. If, however, the Examiner disagrees, the Examiner is invited to contact the undersigned attorney, who will work with the Examiner in a mutual effort to derive satisfactory claim language.

Claims 2-10, 14-22, 25, 27-35, and 40-54 also were rejected under 35 U.S.C. §102(b) as being anticipated by U.S. Patent No. 6,226,618 to *Downs et al.* However, claims 2-10, 14-22, 25, 27-35, and 40-54 are patentably distinguishable over *Downs et al.*, because *Downs et al.* fails to disclose, teach or suggest all of the features recited in the present claims. For example, independent claim 40 (emphasis added) recites:

A method for sharing rights adapted to be associated with an item, the method comprising:

specifying in a first license at least one usage right and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices;

defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item;

defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, wherein **said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights**;

associating at least one state variable with the at least one right in the first license, wherein **the at least one state variable identifies a location where a state of rights is tracked**;

generating in a second license one or more rights based on the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices; and

associating at least one state variable with the at least one right that is shared in the second license, wherein **the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license**.

Independent claim 41 (emphasis added) recites:

A system for sharing rights adapted to be associated with an item, the system comprising:

means for **specifying in a first license at least one usage right and at least one meta-right for the item**, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices;

means for defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item;

means for defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, wherein **said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights**;

means for associating at least one state variable with the at least one right in the first license, wherein **the at least one state variable identifies a location where a state of rights is tracked**;

means for generating in a second license one or more rights based on the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices; and

means for **associating at least one state variable with the at least one right that is shared in the second license**, wherein **the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license**.

Independent claim 42 (emphasis added) recites:

A device for sharing rights adapted to be associated with an item, the device comprising:

means for receiving **a first license specifying at least one usage right and at least one meta-right for the item**, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices, the least one usage right defines a manner of use selected from a plurality of permitted manners of use for the item, the at least one meta-right defines a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, **said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights, at least one state variable is associated with the at least one right in the first license and identifies a location where a state of rights is tracked**; and

means for **generating in a second license one or more rights based on the meta-right in the first license**, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices, **at least one state variable is associated with the at least one right that is shared in the second license, and the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license**.

Thus, the present invention recited in independent claims 40, 41 and 42 includes the novel features of specifying in a first license at least one usage right and at least one meta-right for an item, the at least one meta-right allows one or more users or devices to transfer rights or to derive new rights, associating at least one state variable with the at least one right in the first license, wherein the at least one state variable identifies a location where a state of

rights is tracked, generating in a second license one or more rights based on the meta-right in the first license, and associating at least one state variable with at least one right that is shared in the second license, wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

By contrast, *Downs et al.* is directed to a method and apparatus of securely providing data to a user's system, wherein the data is encrypted so as to only be decryptable by a data decrypting key, the data decrypting key being encrypted using a first public key, and the encrypted data being accessible to the user's system. The method includes transferring the encrypted data decrypting key to a clearing house that possesses a first private key, which corresponds to the first public key; decrypting the data decrypting key using the first private key; re-encrypting the data decrypting key using a second public key; transferring the re-encrypted data decrypting key to the user's system, the user's system possessing a second private key, which corresponds to the second public key; and decrypting the re-encrypted data decrypting key using the second private key. However, *Downs et al.* fails to disclose, teach or suggest at least the noted features recited in independent claims 40, 41 and 42.

Specifically, meta-rights are defined in the present patent application (emphasis added), as rights to “permit granting of rights to others or the derivation of rights” (see, e.g., Applicants' published Specification ¶ [0041]):

[0041] **Rights can specify transfer rights, such as distribution rights, and can permit granting of rights to others or the derivation of rights.** Such rights are referred to as “meta-rights”. **Meta-rights are the rights that one has to [generate], manipulate, modify, [dispose of] or otherwise derive other meta-rights or usage rights. Meta-rights can be thought of as usage rights to usage rights [or other meta-rights].** Meta-rights can include rights to offer, grant, obtain, transfer, delegate, track, surrender, exchange, and revoke usage rights to/from others. Meta-rights can include the rights to modify any of the conditions associated with other rights. For example, a meta-right may be the right to extend or reduce the scope of a particular right. A meta-right may also be the right to extend or reduce the validation period of a right.

As succinctly stated and defined in the present patent application, usage rights permit actions on an item, such as music, video, digital content, and the like. Such actions can include play, read, view, other uses, and the like, of the item. On the other hand, meta-rights permit actions on rights, such as usage rights and meta-rights. Such actions can include generate, modify, transfer, and the like, of rights. By contrast, *Downs et al.* is silent with respect to the novel meta-rights feature of the invention recited in independent claims 40, 41 and 42.

Thus, simply defining “onto what kinds of media the content can be transferred to,” as asserted, at page 5, in the present Office Action, citing column 59, lines 52-54 of *Downs et al.*, fails to disclose, teach or suggest meta-rights, which allow one or more users or devices to transfer rights or to derive new rights, as recited in independent claims 40, 41 and 42.

In addition, the combination of state variables together with meta-rights further patentably distinguishes the invention recited in independent claims 40, 41 and 42 over *Downs et al.* For example, the combination of state variables and meta-rights, advantageously, enables the sharing of rights, wherein one shared right can be derived from another shared right in accordance with a meta-right. In addition, a state variable referring to a location on the device can be used to infer that the right is exclusive to the device, whereas a state variable referring to a location on a server can be used to infer that the right is shared among multiple devices; wherein each device that exercises the right will cause the state variable on the server to be updated. The following two examples illustrate the use of state variables together with meta-rights in more detail (see, e.g., Applicants’ published Specification ¶¶ [0095] and [0099]):

[0095] FIG. 14 is used to illustrate employing of a state variable in deriving inherited usage rights, according to the present invention. In FIG. 14, a derived right can inherit a state variable from meta-rights. For example, a personal computer (PC) of a user, Alice, can be configured to play an e-book according to a license 1403. A personal data assistant (PDA) of Alice also can obtain a right to play the e-book according to offer 1401, if the PC and PDA share the same state variables 1404 and 1405, e.g., “AlicePlayEbook.” A derived right 1402 allows Alice also to play the e-book on her PDA as long as the PDA and the PC share a same count limit 1406 of 5 times.

[0099] FIG. 16 is used to illustrate employing of a state variable in deriving rights that are shared among a dynamic set of rights recipients, according to the present invention. In FIG. 16, an offer 1601 specifies that a distributor can issue site licenses to affiliated clubs, allowing 5 members of each club to concurrently view, play, and the like, content, such as an e-book. A corresponding state variable 1607 associated with such a right can be unspecified in the offer 1601. When corresponding rights 1602 and 1603 are issued to affiliated clubs, the corresponding club identities are used to specify state variables 1608 and 1609 in the issued rights. The offers 1602 and 1603 are meta-rights derived from the offer 1601, with offer being assigned the distinct state variables 1608 and 1609. Further rights 1604-1606 can be derived from the offers 1602 and 1603 to be shared among members of each respective club. The licenses 1604 and 1605 are examples of rights derived from the offer 1602, and which inherit the state variable 1608, e.g., “urn:acme:club,” whereas the license 1606 inherits the state variable 1609, e.g., “urn:foo:club.”

Accordingly, a state variable is not “the number of copies” or “rental terms,” as asserted by the present Office Action, but rather references, for example, a counter or

variable where “the number of copies” or “rental terms” is maintained, and wherein such a counter or variable can be located on a local device or a remote server. The ability to choose the location of a state keeper instead of a specific number, advantageously, provides a mechanism for the rights owner to control rights sharing.

Accordingly, the invention recited in independent claims 40, 41 and 42 and claims dependent therefrom recognizes and solves the following problems:

[0009] However, there are limitations associated with the above-mentioned paradigms wherein only usage rights and conditions associated with content are specified by content owners or other grantors of rights. Once purchased by an end user, a consumer, or a distributor, of content along with its associated usage rights and conditions has no means to be legally passed on to a next recipient in a distribution chain. Further the associated usage rights have no provision for specifying rights to derive other rights, i.e. Rights to modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, revoke, or the like. Common content distribution models often include a multi-tier distribution and usage chain. Known DRM systems do not facilitate the ability to prescribe rights and conditions for all participants along a content distribution and usage chain. Therefore, it is difficult for a content owner to commercially exploit content unless the owner has a relationship with each party in the distribution chain.

The invention recited in independent claims 40, 41 and 42 and claims dependent therefrom provides the following advantages:

[0090] There are multiple ways to specify the scope of state variables, each of which can affect whether the derivative state variables can be shared, how the derivative state variables can be shared, and the like. For example, a state variable can be local, and solely confined to a recipient or can be global, and shared by a predetermined group of recipients. A global state variable can be shared by a group of recipients not determined when derived rights are issued, but to be specified later, perhaps based on certain rules defined in the license or based on other means. A global state variable can be shared between one or more rights suppliers, predetermined recipients, un-specified recipients, and the like. Advantageously, depending on the sharing employed with a given a business model and the rights granted in the meta-rights, state variables can be created at different stages of the value chain.

By contrast, *Downs et al.* fails disclose, teach or suggest the noted features, fails to recognize or solve the noted problems, and fails to provide the advantages of the invention recited in independent claims 40, 41 and 42.

In addition, new claim 55 is added to recite that the novel feature, wherein rights can be derived from a second license, as compared to *Downs et al.* that only deals with a first license. New claims 56 and 57 are added to recite some novel ways to use a state variable.

Accordingly, the dependent claims are allowable on their own merits and for at least the reasons as argued above with respect to independent claims 40, 41 and 42.

In view of the foregoing, it is submitted that the present application is in condition for allowance and a notice to that effect is respectfully requested. However, if the Examiner deems that any issue remains after considering this response, the Examiner is invited to contact the undersigned attorney to expedite the prosecution and engage in a joint effort to work out a mutually satisfactory solution.

Respectfully submitted,

NIXON PEABODY, LLP

/Carlos R. Villamar, Reg. # 43,224/

Carlos R. Villamar

Reg. No. 43,224

NIXON PEABODY LLP
CUSTOMER NO.: 22204
401 9th Street, N.W., Suite 900
Washington, DC 20004
Tel: 202-585-8000
Fax: 202-585-8080

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:) Confirmation No.: 8299
Mai NGUYEN, *et al.*) Group Art Unit: 3621
Serial No. 10/956,070) Examiner: Evens J. Augustin
Filed: October 4, 2004)
For: **SYSTEM AND METHOD FOR**)
RIGHTS OFFERING AND
GRANTING USING SHARED STATE
VARIABLES

U.S. Patent and Trademark Office
Customer Window, Mail Stop Non-Fee Amendment
Randolph Building
401 Dulany Street
Alexandria, VA 22314

AMENDMENT AFTER NON-FINAL REJECTION

Sir:

In response to the non-final Office Action mailed **November 30, 2006**, the below amendments and following remarks are submitted in connection with the above-identified application.

Amendments to the Claims:

1. (Cancelled)

2. (Previously presented) The method of claim 40, wherein the state variable in the first or second license inherits a state thereof for content usage or rights derivation from other generated usage rights and meta-rights.

3. (Previously presented) The method of claim 40, wherein the state variable in the first or second license shares a state thereof for content usage or rights derivation with other generated usage rights and meta-rights.

4. (Previously presented) The method of claim 40, wherein the state variable in the first or second license inherits a remaining state for content usage or rights derivation from other generated usage rights and meta-rights.

5. (Previously presented) The method of claim 40, wherein the state variable in the first or second license is updated upon exercise of a right associated with the state variable.

6. (Previously presented) The method of claim 40, wherein the state variable in the first or second license represents a collection of states.

7. (Previously presented) The method of claim 40, further comprising:
generating in a third license one or more rights from at least one of the usage right and the meta-right in the second license,

wherein the one or more rights in the third license includes at least one right that is shared among one or more users or devices;

associating at least one state variable with the at least one right that is shared in the third license,

wherein the at least one state variable that is associated with the third license is based on the at least one state variable that is associated with the second license.

8. (Previously presented) The method of claim 40, further comprising a plurality of state variables that determine the state of the at least one right that is shared in the first or the second license.

9. (Cancelled)

10. (Previously presented) The method of claim 40, wherein the state variable in the second license is transferred from the at least one right in the first license and is associated with the right that is shared in the second license.

11-13. (Cancelled)

14. (Previously presented) The system of claim 41, wherein the state variable in the first or second license inherits a state thereof for content usage or rights derivation from other generated usage rights and meta-rights.

15. (Previously presented) The system of claim 41, wherein the state variable in the first or second license shares a state thereof for content usage or rights derivation with other generated usage rights and meta-rights.

16. (Previously presented) The system of claim 41, wherein the state variable in the first or second license inherits a remaining state for content usage or rights derivation from other generated usage rights and meta-rights.

17. (Previously presented) The system of claim 41, wherein the state variable in the first or second license is updated upon exercise of a right associated with the state variable.

18. (Previously presented) The system of claim 41, wherein the state variable in the first or second license represents a collection of states.

19. (Previously presented) The system of claim 41, further comprising:
means for generating in a third license one or more rights from at least one of the usage right and the meta-right in the second license,

wherein the one or more rights in the third license includes at least one right that is shared among one or more users or devices;

means for associating at least one state variable with the at least one right that is shared in the third license,

wherein the at least one state variable that is associated with the third license is based on the at least one state variable that is associated with the second license.

20. (Previously presented) The system of claim 41, including a plurality of state variables that determine the state of the at least one right that is shared in the first or the second license.

21. (Cancelled)

22. (Previously presented) The system of claim 41, wherein the state variable in the second license is transferred from the at least one right in the first license and is associated with the right that is shared in the second license.

23-24. (Cancelled)

25. (Previously presented) The system of claim 41, wherein the system is implemented with one or more hardware and software components.

26. (Cancelled)

27. (Previously presented) The device of claim 42, wherein the state variable in the first or second license inherits a state thereof for content usage or rights derivation from other generated usage rights and meta-rights.

28. (Previously presented) The device of claim 42, wherein the state variable in the first or second license shares a state thereof for content usage or rights derivation with other generated usage rights and meta-rights.

29. (Previously presented) The device of claim 42, wherein the state variable in the first or second license inherits a remaining state for content usage or rights derivation from other generated usage rights and meta-rights.

30. (Previously presented) The device of claim 42, wherein the state variable in the first or second license is updated upon exercise of a right associated with the state variable.

31. (Previously presented) The device of claim 42, wherein the state variable in the first or second license represents a collection of states.

32. (Previously presented) The device of claim 42, wherein a third license includes one or more rights from at least one of the usage right and the meta-right in the second license,

the one or more rights in the third license includes at least one right that is shared among one or more users or devices,

at least one state variable is associated with the at least one right that is shared in the third license, and

the at least one state variable that is associated with the third license is based on the at least one state variable that is associated with the second license.

33. (Previously presented) The device of claim 42, including a plurality of state variables that determine the state of the at least one right that is shared in the first or the second license.

34. (Cancelled)

35. (Previously presented) The device of claim 42, wherein the state variable in the second license is transferred from the at least one right in the first license and is associated with the right that is shared in the second license.

36-39. (Cancelled)

40. (Currently amended) A method for sharing rights adapted to be associated with an item, the method comprising:

specifying in a first license at least one usage right and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices;

defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item;

defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, wherein said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights;

associating at least one state variable with the at least one right in the first license, wherein the at least one state variable ~~is shared among the one or more users or devices, and is specified as unspecified or is assigned an identification in said first license, and said identification references~~ identifies a location where a state of rights is tracked;

generating in a second license one or more rights based on the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices; and

associating at least one state variable with the at least one right that is shared in the second license, wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

41. (Currently amended) A system for sharing rights adapted to be associated with an item, the system comprising:

means for specifying in a first license at least one usage right and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices;

means for defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item;

means for defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, wherein said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights;

means for associating at least one state variable with the at least one right in the first license, wherein the at least one state variable ~~is shared among the one or more users or devices, and is specified as unspecified or is assigned an identification in said first license, and said identification references~~ identifies a location where a state of rights is tracked;

means for generating in a second license one or more rights based on the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices; and

means for associating at least one state variable with the at least one right that is shared in the second license, wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

42. (Currently amended) A device for sharing rights adapted to be associated with an item, the device comprising:

means for receiving a first license specifying at least one usage right and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices, the least one usage right defines a manner of use selected from a plurality of permitted manners of use for the item, the at least one meta-right defines a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights, at least one state variable is associated with the at least one right in the first license and ~~is shared among the one or more users or devices and is specified as unspecified or is assigned an identification in said first license, and said identification references~~ identifies a location where a state of rights is tracked; and

means for generating in a second license one or more rights based on the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices, at least one state variable is associated with the at least one right that is shared in the second license, and the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

43. (Previously presented) The method of claim 40, wherein the method is implemented with one or more hardware and software components configured to perform the steps of the method.

44. (Previously presented) The method of claim 40, wherein the method is implemented with one or more computer readable instructions embedded on a computer readable medium and configured to cause one or more computer processors to perform the steps of the method.

45. (Previously presented) The device of claim 42, wherein the device is implemented with one or more hardware and software components.

46-48. (Cancelled)

49. (Previously presented) The method of claim 40, wherein the plurality of permitted manners of use for the item include copy, transfer, loan, play, print, delete, extract, embed, edit, authorize, install, and un-install the item.

50. (Previously presented) The system of claim 41, wherein the plurality of permitted manners of use for the item include copy, transfer, loan, play, print, delete, extract, embed, edit, authorize, install, and un-install the item.

51. (Previously presented) The device of claim 42, wherein the plurality of permitted manners of use for the item include copy, transfer, loan, play, print, delete, extract, embed, edit, authorize, install, and un-install the item.

52. (Previously presented) The method of claim 40, wherein the plurality of permitted manners of rights derivation for the item include issue, modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, and revoke rights for the item.

53. (Previously presented) The system of claim 41, wherein the plurality of permitted manners of rights derivation for the item include issue, modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, and revoke rights for the item.

54. (Previously presented) The device of claim 42, wherein the plurality of permitted manners of rights derivation for the item include issue, modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, and revoke rights for the item.

55. (New) The method of claim 40, further comprising:

generating in a third license one or more rights based on the meta-right in the second license, wherein the one or more rights in the third license includes at least one right that is shared among one or more users or devices; and

associating at least one state variable with the at least one right that is shared in the third license, wherein the at least one state variable that is associated with the third license is based on the at least one state variable that is associated with the second license.

56. (New) The method of claim 40, wherein the at least one state variable that is associated with the second license is the same as the at least one state variable that is associated with the first license, if the at least one state variable that is associated with the first license does not identify an unspecified location.

57. (New) The method of claim 40, wherein the at least one state variable that is associated with the second license is assigned a new location identification, if the at least one state variable that is associated with the first license identifies an unspecified location.

Electronic Acknowledgement Receipt

EFS ID:	1554578
Application Number:	10956070
International Application Number:	
Confirmation Number:	8299
Title of Invention:	System and method for rights offering and granting using shared state variables
First Named Inventor/Applicant Name:	Mai Nguyen
Customer Number:	22204
Filer:	Carlos Rafael Villamar/Kisha Avery
Filer Authorized By:	Carlos Rafael Villamar
Attorney Docket Number:	111325-235000
Receipt Date:	28-FEB-2007
Filing Date:	04-OCT-2004
Time Stamp:	19:23:26
Application Type:	Utility

Payment information:

Submitted with Payment	no
------------------------	----

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)	Multi Part /.zip	Pages (if appl.)
1	Amendment - After Non-Final Rejection	111325_235000_US_Respo nse_to_30_NOV_06_NFOA. pdf	71328	no	15

Warnings:

Information:	
Total Files Size (in bytes):	71328
<p>This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.</p> <p><u>New Applications Under 35 U.S.C. 111</u> If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.</p> <p><u>National Stage of an International Application under 35 U.S.C. 371</u> If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.</p> <p><u>New International Application Filed with the USPTO as a Receiving Office</u> If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.</p>	

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD Substitute for Form PTO-875	Application or Docket Number 10/956,070	Filing Date 10/04/2004	<input type="checkbox"/> To be Mailed
---	---	----------------------------------	---------------------------------------

APPLICATION AS FILED – PART I			OTHER THAN SMALL ENTITY				
(Column 1)		(Column 2)	SMALL ENTITY <input type="checkbox"/>		OR	SMALL ENTITY	
FOR	NUMBER FILED	NUMBER EXTRA	RATE (\$)	FEE (\$)		RATE (\$)	FEE (\$)
<input type="checkbox"/> BASIC FEE <small>(37 CFR 1.16(a), (b), or (c))</small>	N/A	N/A	N/A		OR	N/A	
<input type="checkbox"/> SEARCH FEE <small>(37 CFR 1.16(k), (l), or (m))</small>	N/A	N/A	N/A			N/A	
<input type="checkbox"/> EXAMINATION FEE <small>(37 CFR 1.16(o), (p), or (q))</small>	N/A	N/A	N/A			N/A	
TOTAL CLAIMS <small>(37 CFR 1.16(i))</small>	minus 20 =	*	X \$ =			X \$ =	
INDEPENDENT CLAIMS <small>(37 CFR 1.16(h))</small>	minus 3 =	*	X \$ =			X \$ =	
<input type="checkbox"/> APPLICATION SIZE FEE <small>(37 CFR 1.16(s))</small>	If the specification and drawings exceed 100 sheets of paper, the application size fee due is \$250 (\$125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
<input type="checkbox"/> MULTIPLE DEPENDENT CLAIM PRESENT <small>(37 CFR 1.16(j))</small>							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II					OTHER THAN SMALL ENTITY				
(Column 1)		(Column 2)	(Column 3)		SMALL ENTITY		OR	SMALL ENTITY	
AMENDMENT	02/28/2007	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE (\$)	ADDITIONAL FEE (\$)		RATE (\$)	ADDITIONAL FEE (\$)
	Total <small>(37 CFR 1.16(i))</small>	* 39	Minus	** 39 = 0	X \$ =		OR	X \$50=	0
	Independent <small>(37 CFR 1.16(h))</small>	* 3	Minus	***3 = 0	X \$ =		OR	X \$200=	0
	<input type="checkbox"/> Application Size Fee <small>(37 CFR 1.16(s))</small>								
	<input type="checkbox"/> FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <small>(37 CFR 1.16(j))</small>						OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	0

APPLICATION AS AMENDED – PART II					OTHER THAN SMALL ENTITY				
(Column 1)		(Column 2)	(Column 3)		SMALL ENTITY		OR	SMALL ENTITY	
AMENDMENT	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE (\$)	ADDITIONAL FEE (\$)		RATE (\$)	ADDITIONAL FEE (\$)	
	Total <small>(37 CFR 1.16(i))</small>	*	Minus	** =	X \$ =		OR	X \$ =	
	Independent <small>(37 CFR 1.16(h))</small>	*	Minus	*** =	X \$ =		OR	X \$ =	
	<input type="checkbox"/> Application Size Fee <small>(37 CFR 1.16(s))</small>								
	<input type="checkbox"/> FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <small>(37 CFR 1.16(j))</small>						OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
 ** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
 *** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
 The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
 Jacqueline E. Couplin

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/956,070	10/04/2004	Mai Nguyen	111325-235000	8299
22204	7590	05/21/2007	EXAMINER	
NIXON PEABODY, LLP 401 9TH STREET, NW SUITE 900 WASHINGTON, DC 20004-2128			AUGUSTIN, EVENS J	
			ART UNIT	PAPER NUMBER
			3621	
			MAIL DATE	DELIVERY MODE
			05/21/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

DETAILED ACTION

Acknowledgements

1. This is in response to an amendment filed on 28 February 2007. Claims 40-42 have been amended. Claims 55-57 have been added. Claims 2-8, 10, 14-20, 22, 25, 27-33, 35, 40-45, and 49-57 are pending.

Response to Arguments

2. The United States Patent and Trademark Office has fully considered the applicant's arguments filed on 28 February 2007, but has not found those arguments to be persuasive.

Argument 1: Downs et al., fails to disclose, teach or suggest meta-rights, which allow one or more users or devices to transfer rights or to derive new rights.

Response 1: According to the applicant's specification, meta-rights are the rights that one has to manipulate, modify, or otherwise derive other meta-rights or usage rights. Meta-rights can be thought of as usage rights to usage rights (page 9, par. 41). Content providers (entity(s) that supplies the content), providing (equivalent to generating) usage conditions (equivalent to usage rights) also stipulate that the content stores or distributors also have rights to add or narrow the original usage rights (meta-rights or rights derived from the initial usage rights) (column 21, lines 30-36).

Additionally, state variables can be the number of copies a user is allowed to make (column 59, line 50 or rental terms (column 59, lines 55-60). Content providers and distributors specify the number of plays and local copies allowed for the Content, and whether or not the Content may be recorded to an external portable device (state variable). Downs et al. keep track of the content's copy/play usage and update the copy/play status (column 20, lines 43-50, column

12, lines 11-12). The system uses watermarks to ensure that the content is being played in a compliant user device (col. 7, lines 45-55). Inherently the identity or location of where the content is being played or copied has to be established, in order to determine whether or not a user is compliant.

Application stands finally rejected.

Claim Interpretation

3. In determining patentability of an invention over the prior art, the USPTO has considered all claimed limitations, and interpreted as broadly as their terms reasonably allow. Additionally, all words in the claims have been considered in judging the patentability of the claims against the prior art.

4. It should also be noted that, in the office action that:

A. Items in the rejection that are in quotation marks are claimed language/limitations

B. Functional recitation(s) using the word “for” or other functional terms (*e.g.* “...for content usage or rights derivation from other generated usage rights and meta-rights” as recited in claims 2-4) have been considered but given less patentable weight¹

because they fail to add any steps and are thereby regarded as intended use language.

To be especially clear, the Examiner has considered all claim limitations. However a recitation of the intended use of the claimed invention must result in additional steps.

See *Bristol-Myers Squibb Co. v. Ben Venue Laboratories, Inc.*, 246 F.3d 1368, 1375-76, 58 USPQ2d 1508, 1513 (Fed. Cir. 2001) (Where the language in a method claim

¹ See *e.g. In re Gulack*, 703 F.2d 1381, 217 USPQ 401, 404 (Fed. Cir. 1983)(stating that although all limitations must be considered, not all limitations are entitled to patentable weight).

states only a purpose and intended result, the expression does not result in a manipulative difference in the steps of the claim.).

- C. Word(s) that are separated by “/” are being examined as being synonymous or equivalent
- D. The USPTO interprets claim limitations that contain statement(s) such as “*if, may, might, can, could, when, potentially, possibly*”, as optional language (this list of examples is not intended to be exhaustive). As matter of linguistic precision, optional claim elements do not narrow claim limitations, since they can always be omitted (*In re Johnston*, 77 USPQ2d 1788 (Fed. Circ. 2006)). They will be given less patentable weight, because language that suggests or makes optional but does not require steps to be performed or does not limit a claim to a particular structure does not limit the scope of a claim or claim limitation.
- E. Independent claims are examined together, since they are not patentable distinct. If applicant expressly states on the record that two or more independent and distinct inventions are claimed in a single application, the Examiner may require the applicant to elect an invention to which the claims will be restricted.
- F. Any official notices taken by the USPTO that are not adequately traversed by applicant will be taken to be admitted prior art.

Claim Rejections - 35 USC § 112 – 1st Paragraph

5. The following is a quotation of the first paragraph of 35 U.S.C. 112:
-

The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

6. Claim 55 is rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claims contain subject matter, which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

7. The purpose of the requirement that the specification describe the invention in such terms that one skilled in the art can make and use the claimed invention is to ensure that the invention is communicated to the interested public in a meaningful way. The information contained in the disclosure of an application must be sufficient to inform those skilled in the relevant art how to both make and use the claimed invention.

8. Any analysis of whether a particular claim is supported by the disclosure in an application requires a determination of whether that disclosure, when filed, contained sufficient information regarding the subject matter of the claims as to enable one skilled in the pertinent art to make and use the claimed invention. The standard for determining whether the specification meets the enablement requirement was cast in the Supreme Court decision of *Mineral Separation v. Hyde*, 242 U.S. 261, 270 (1916) which postured the question: is the experimentation needed to practice the invention undue or unreasonable? That standard is still the one to be applied. In *re Wands*, 858 F.2d 731, 737, 8 USPQ2d 1400, 1404 (Fed. Cir. 1988). Accordingly, even though the statute does not use the term “undue experimentation,” it has been interpreted to require that the claimed invention be enabled so that any person skilled in the art can make and use the invention without undue experimentation. In *re Wands*, 858 F.2d at 737, 8 USPQ2d at 1404

(Fed. Cir. 1988). See also *United States v. Teletronics, Inc.*, 857 F.2d 778, 785, 8 USPQ2d 1217, 1223 (Fed. Cir. 1988) (“The test of enablement is whether one reasonably skilled in the art could make or use the invention from the disclosures in the patent coupled with information known in the art without undue experimentation.”).

9. As per claim 55, it recite the steps of “generating in a third license one or more rights based on the meta-right in the second license, wherein the one or more rights in the third license includes at least one right that is shared among one or more users or devices”. This limitation is considered as new matter since it is not disclosed in the specification.

Claim Rejections - 35 USC § 102

10. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

A person shall be entitled to a patent unless –

(a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States. . . .

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

11. Claims 2-10, 14-22, 25, 27-35, and 40-54 are rejected under 35 U.S.C. 102(b) as being anticipated by Downs et al (U.S 6226618).

12. As per claims 2-10, 14-22, 25, 27-35, and 40-54, Downs et al. disclose an invention that broadly relates to the field of electronic commerce and more particularly to a system and related tools for the secure delivery and rights management of digital assets, such as print media, films, games, and music over global communications networks such as the Internet and the World Wide Web. The invention includes the means and devices (hardware and software combination) (columns 53, lines 65-67, column 54, lines 1-3) to accomplish the items below- Claims 25, 38, 39, 41-45, 48, 51, 54. The invention comprising of the following:

- A. Owners setting/specifying initial usage rights/licensing (**first license**) for content to the distributors (column 21, lines 30-33) ("**specifying in a first license at least one usage right and at least one meta-right for the item**") – *Claims 40-42*
- B. Example of usage rights include (column 59, lines 38-69 ("**specifying in a first license at least one usage right and at least one meta-right for the item**")), ("**defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item**")– *Claims 40, 49-54:*

Compressed Version	384 Kbps
Type of user	Private Consumer
Type of Transaction	Purchase or Rental
Number Of copies	1
Rental Terms	14 Days
Transfer on What Media	Mini Disc or Computer

- C. Owners setting initial usage rights/licensing (**first license**) for content to the distributors (column 21, lines 30-33). Those usage rights can be modified by the

digital store (column 21, lines 33-39) to create **secondary licensing** or customized licensing (column 10, lines 15-18) to the end user – ("**wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices**") *Claims 40-42, 10, 22, 35*

- D. Content providers (entity that supplies the content), providing (equivalent to generating) usage conditions (equivalent to usage rights), The content providers also stipulate that the content stores or distributors can add or narrow the original usage rights (meta-rights or rights derived from the initial usage rights) (column 21, lines 30-36) - *Claims 40-42*
- E. Usage rights being copy restrictions, which is manner in which the content can be used (column 9, 32-34, col. 26, lines 10-12). - *Claims 40-42*
- F. Content stores or distributors can add or narrow the original usage rights (sub-rights) (column 21, lines 30-36) - *Claims 40-42*
- G. The system also defines the manner in which the content can be used (**meta-rights**) such as onto what kinds of media the content can be transferred to (column 59, lines 52-54) – *Claims 40-42, 46-48*
- H. State variable such the number of copies a user is allowed to make (column 59, line 50 or rental terms (column 59, lines 55-60) - *Claims 40-42*
- I. State variables can be the number of copies a user is allowed to make - ("**associating at least one state variable with the at least one right** ") (column 59, line 50 or rental terms (column 59, lines 55-60). Content providers and distributors specify the number of plays and local copies allowed for the Content, and whether or not the Content may be recorded to an external portable device (state variable). Downs et al.

keep track of the content's copy/play usage and update the copy/play status (column 20, lines 43-50, column 12, lines 11-12). The system also uses watermarks, as state variable, to ensure that the content is being played in a compliant user device (col. 7, lines 45-55). Inherently the identity or location of where the content is being played or copied has to be established, in order to determine whether or not a user is compliant. - (**"defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, wherein said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights"**) -*Claims 40-42, 55-57*

- J. The secondary licensing such as restrictions on rental time period can not violate the initial time period set by the initial licensing (column 21, line 35) - (**"generating in a second license one or more rights based on the meta-right in the first license"**)
Claims 40-42
- K. The state variable is derived from the usage rights (column 59, line 50) or rental terms (column 59, lines 55-60) - *Claims 2-4, 14-16, 27-29*
- L. The system keeps track of the content's copy/play usage and updates the copy/play status (column 20, lines 49-50) – *Claims 5, 17, 30*
- M. A state variable can represent various other states, for example an item that rented can affect the number of copies that can be made or whether or not copies can be made -
Claims 6, 8, 18, 20, 31,33
- N. The system embeds a code on every copy the content, as it is transferred form user device to the next. When the Digital Content is accessed in a compliant End-User Devices, the End-User Player Application reads the watermark to check the use

restrictions and updates the watermark as required. If the requested use of the content does not comply with the usage conditions, e.g., the number of copies has been exhausted, the End-User Device(s) will not perform the request (column 7, lines 40-55) - *Claims 7, 19, 32*

- O. The content does not specify how the initial set of rights and variable are to modified, as long as it does not violate the initial licensing (column 21, line 35) - *Claims 9, 21, 34*

Conclusion

13. **THIS ACTION IS MADE FINAL.** Any new ground(s) of rejection is due to the applicant's amendment. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

14. A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

15. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Evens Augustin whose telephone number is 571-272-6860. The examiner can normally be reached on Monday thru Friday 8 to 5 pm.

16. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Andrew Fischer can be reached on 571-272-6779.



Evens J. Augustin
May 11, 2007
Art Unit 3621



ANDREW J. FISCHER
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 3600

Index of Claims



Application/Control No.

10/956,070

Examiner

Evens Augustin

Applicant(s)/Patent under Reexamination

NGUYEN ET AL.

Art Unit

3621

√	Rejected
=	Allowed

-	(Through numeral) Cancelled
+	Restricted

N	Non-Elected
I	Interference

A	Appeal
O	Objected

Claim		Date					
Final	Original	5/11/07					
	1	-					
	2	√					
	3	√					
	4	√					
	5	√					
	6	√					
	7	√					
	8	√					
	9	-					
	10	√					
	11	-					
	12	-					
	13	-					
	14	√					
	15	√					
	16	√					
	17	√					
	18	√					
	19	√					
	20	√					
	21	-					
	22	√					
	23	-					
	24	-					
	25	√					
	26	-					
	27	√					
	28	√					
	29	√					
	30	√					
	31	√					
	32	√					
	33	√					
	34	-					
	35	√					
	36	-					
	37	-					
	38	-					
	39	-					
	40	√					
	41	√					
	42	√					
	43	√					
	44	√					
	45	√					
	46	-					
	47	-					
	48	-					
	49	√					
	50	√					

Claim		Date					
Final	Original	5/11/07					
	51	√					
	52	√					
	53	√					
	54	√					
	55	√					
	56	√					
	57	√					
	58						
	59						
	60						
	61						
	62						
	63						
	64						
	65						
	66						
	67						
	68						
	69						
	70						
	71						
	72						
	73						
	74						
	75						
	76						
	77						
	78						
	79						
	80						
	81						
	82						
	83						
	84						
	85						
	86						
	87						
	88						
	89						
	90						
	91						
	92						
	93						
	94						
	95						
	96						
	97						
	98						
	99						
	100						

Claim		Date					
Final	Original						
	101						
	102						
	103						
	104						
	105						
	106						
	107						
	108						
	109						
	110						
	111						
	112						
	113						
	114						
	115						
	116						
	117						
	118						
	119						
	120						
	121						
	122						
	123						
	124						
	125						
	126						
	127						
	128						
	129						
	130						
	131						
	132						
	133						
	134						
	135						
	136						
	137						
	138						
	139						
	140						
	141						
	142						
	143						
	144						
	145						
	146						
	147						
	148						
	149						
	150						

REQUEST FOR CONTINUED EXAMINATION(RCE)TRANSMITTAL (Submitted Only via EFS-Web)

Application Number	10956070	Filing Date	2004-10-04	Docket Number (if applicable)	111325-235000	Art Unit	3621
First Named Inventor	Mai NGUYEN			Examiner Name	Evens J. Augustin		

This is a Request for Continued Examination (RCE) under 37 CFR 1.114 of the above-identified application.

Request for Continued Examination (RCE) practice under 37 CFR 1.114 does not apply to any utility or plant application filed prior to June 8, 1995, or to any design application. The Instruction Sheet for this form is located at WWW.USPTO.GOV

SUBMISSION REQUIRED UNDER 37 CFR 1.114

Note: If the RCE is proper, any previously filed unentered amendments and amendments enclosed with the RCE will be entered in the order in which they were filed unless applicant instructs otherwise. If applicant does not wish to have any previously filed unentered amendment(s) entered, applicant must request non-entry of such amendment(s).

Previously submitted. If a final Office action is outstanding, any amendments filed after the final Office action may be considered as a submission even if this box is not checked.

Consider the arguments in the Appeal Brief or Reply Brief previously filed on _____

Other _____

Enclosed

Amendment/Reply

Information Disclosure Statement (IDS)

Affidavit(s)/ Declaration(s)

Other _____

MISCELLANEOUS

Suspension of action on the above-identified application is requested under 37 CFR 1.103(c) for a period of months _____
(Period of suspension shall not exceed 3 months; Fee under 37 CFR 1.17(i) required)

Other _____

FEES

The RCE fee under 37 CFR 1.17(e) is required by 37 CFR 1.114 when the RCE is filed.

The Director is hereby authorized to charge any underpayment of fees, or credit any overpayments, to Deposit Account No 192380

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT REQUIRED

Patent Practitioner Signature

Applicant Signature

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Signature of Registered U.S. Patent Practitioner			
Signature	/Stephen M. Hertzler/	Date (YYYY-MM-DD)	2007-10-31
Name	Stephen M. Hertzler	Registration Number	58247

This collection of information is required by 37 CFR 1.114. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these records.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Electronic Patent Application Fee Transmittal

Application Number:	10956070			
Filing Date:	04-Oct-2004			
Title of Invention:	System and method for rights offering and granting using shared state variables			
First Named Inventor/Applicant Name:	Mai Nguyen			
Filer:	Stephen M. Hertzler			
Attorney Docket Number:	111325-235000			
Filed as Large Entity				
Utility Filing Fees				
Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Extension - 3 months with \$0 paid	1253	1	1050	1050

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
Miscellaneous:				
Request for continued examination	1801	1	810	810
Total in USD (\$)				1860

REMARKS

Claims 2-8, 10, 14-20, 22, 25, 27-33, 35, 40-45, and 49-57 are pending in this application. By this amendment, claim 55 is amended. Thus, claims 2-8, 10, 14-20, 22, 25, 27-33, 35, 40-45, and 49-57 remain pending in this application. Reconsideration and allowance of this application is respectfully requested.

Claim 55 stands rejected under 35 U.S.C. § 112, first paragraph, as failing to comply with the enablement requirement. In particular, the Examiner asserts that the steps of “generating in a third license one or more rights based on the meta-right in the second license, wherein the one or more rights in the third license includes at least one right that is shared among one or more users or devices” is not disclosed in the specification.

However, Applicants respectfully submit that amended claim 55, which recites, in relevant part, “generating in a further license one or more rights based on the meta-right in the second license, wherein the one or more rights in the further license includes at least one right that is shared among one or more users or devices; and associating at least one state variable with the at least one right that is shared in the further license, wherein the at least one state variable that is associated with the further license is based on the at least one state variable that is associated with the second license,” is fully supported by the specification.

For example, according to the invention, the first license is the offer allowing a distributor to issue licenses for affiliated clubs; the distributor derives from the offer a second license allowing an affiliated club to issue licenses to its member; and the club derives a further license (i.e. a third license) allowing a member to play the ebook.

In particular, paragraph [0099] of the published application provides:

FIG. 16 is used to illustrate employing of a state variable in deriving rights that are shared among a dynamic set of rights recipients, according to the present invention. In FIG. 16, an offer 1601 specifies that a distributor can issue site licenses to affiliated clubs, allowing 5 members of each club to concurrently view, play, and the like, content, such as an e-book. A corresponding state variable 1607 associated with such a right can be unspecified in the offer 1601. When corresponding rights 1602 and 1603 are issued to affiliated clubs, the corresponding club identities are used to specify state variables 1608 and 1609 in the issued rights. The offers 1602 and 1603 are meta-rights derived from the offer 1601, with offer being assigned the distinct state variables 1608 and 1609. Further rights 1604-1606 can be derived from the offers 1602 and 1603 to be shared among members of each respective club. The licenses 1604 and 1605 are examples

of rights derived from the offer 1602, and which inherit the state variable 1608, e.g., "urn:acme:club," whereas the license 1606 inherits the state variable 1609, e.g., "urn:foo:club."

For reference, Figure 16 is shown below:

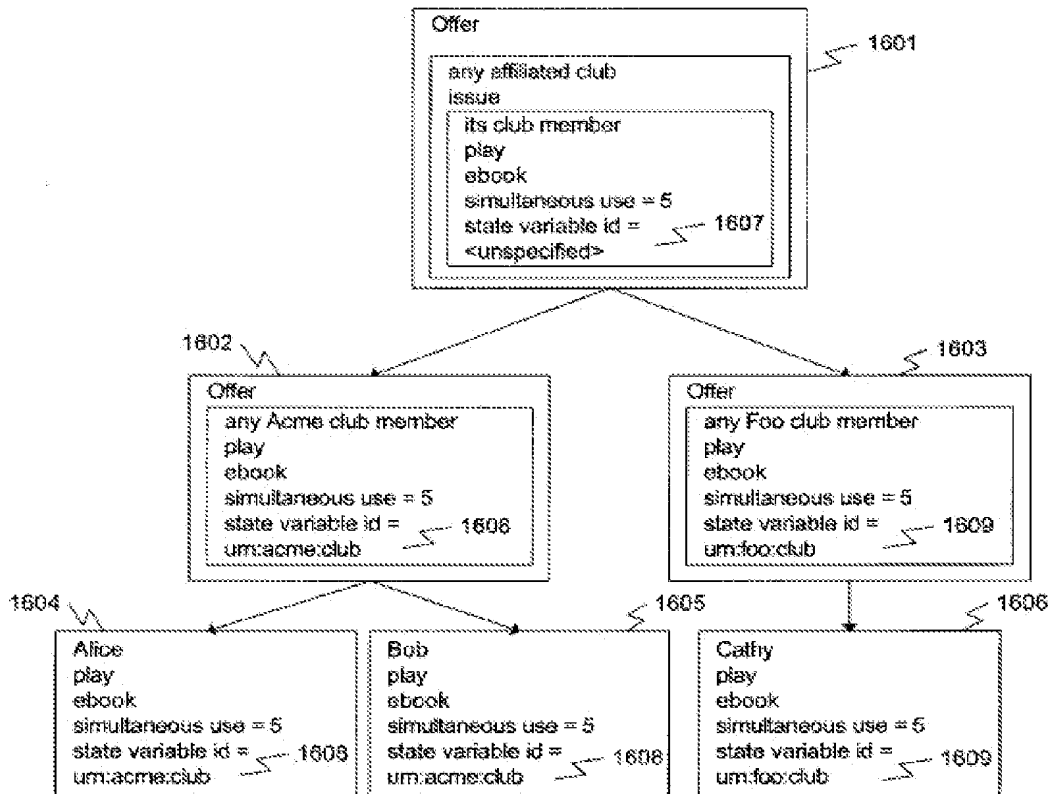


Fig. 16

In addition, paragraph [0101] of the published application provides:

The state of rights can depend on more than one state variable. FIG. 18 is used to illustrate employing of multiple state variables to represent one state of rights, according to the present invention. The example described with respect to FIG. 18 builds upon the example described with respect to FIG. 16. In FIG. 18, a usage right can be tracked by employing multiple state variables 1807 and 1808 in an offer 1801. The state variable 1808, for example, representing a priority level, can stay unspecified in the corresponding offers 1802 and 1803 (e.g., site licenses). The corresponding state variables 1809-1811, for example, used for setting a priority, can be assigned to each member in the corresponding licenses 1804, 1805 and 1806. The corresponding right to view, play, and the like, can now be dependent on two state variables, effectively restricting 5 simultaneous views, plays, and the like, per priority level.

For reference, Figure 18 is shown below:

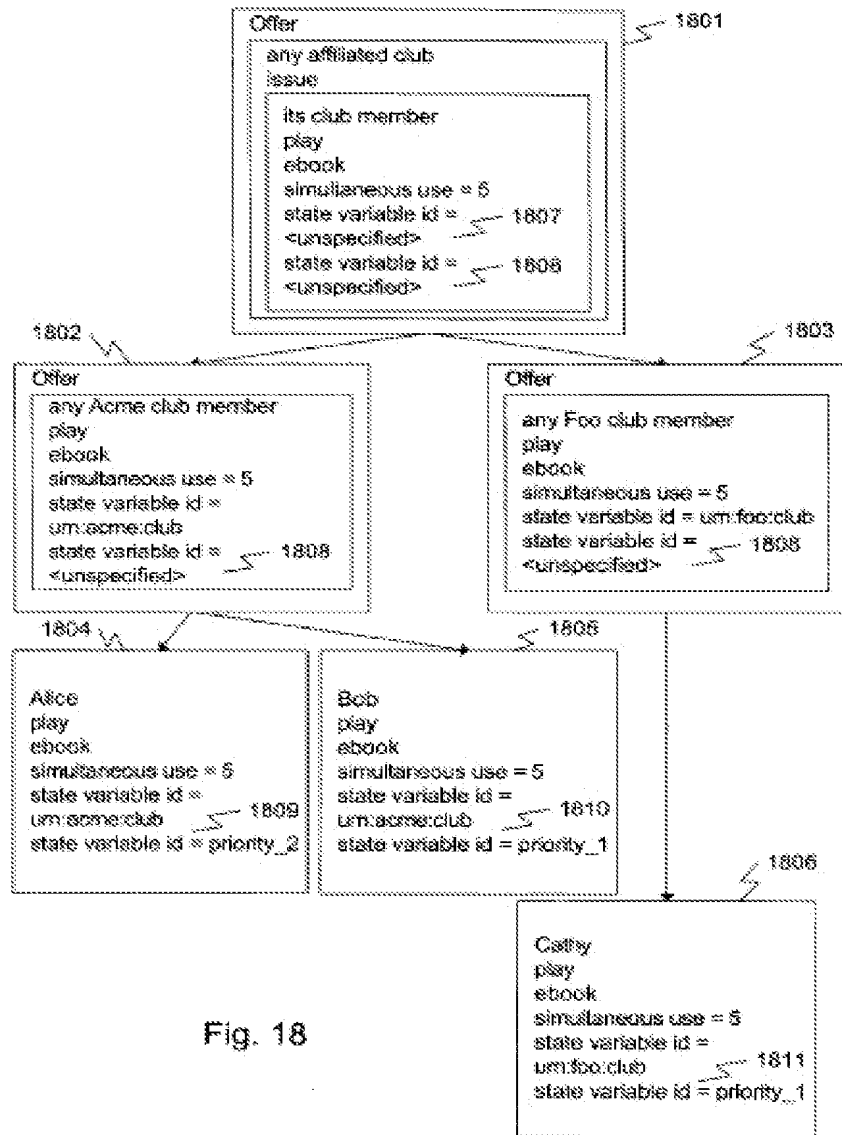


Fig. 18

Thus, the features recited in claim 55, as presented herein, are fully and clearly supported by at least Figs. 16 and 18, and related discussion in at least paragraphs [0099] and [0101] of the published application. Accordingly, Applicants respectfully submit that the rejection of claim 55 under 35 U.S.C. § 112 should be reconsidered and withdrawn.

Claims 2-10, 14-22, 25, 27-35, and 40-54 stand rejected under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent No. 6,226,618 to Downs et al. However, Applicants

respectfully submit that Downs fails to disclose, suggest, or render obvious, each and every feature recited in claims 2-10, 14-22, 25, 27-35, and 40-54.

For example, independent claim 40 recites a method for sharing rights adapted to be associated with an item, the method comprising *specifying in a first license at least one usage right and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices*, defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item, defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, wherein *the at least one meta-right allows the one or more users or devices to transfer rights or to derive new rights*, associating at least one state variable with the at least one right in the first license, wherein *the at least one state variable identifies a location where a state of rights is tracked*, generating in a second license one or more rights based on the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices, and *associating at least one state variable with the at least one right that is shared in the second license, wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.*

In addition, independent claim 41 recites a system for sharing rights adapted to be associated with an item, the system comprising means for *specifying in a first license at least one usage right and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices*, means for defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item, means for defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, wherein *the at least one meta-right allows the one or more users or devices to transfer rights or to derive new rights*, means for associating at least one state variable with the at least one right in the first license, wherein *the at least one state variable identifies a location where a state of rights is tracked*, means for generating in a second license one or more rights based on the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more

users or devices, and means for *associating at least one state variable with the at least one right that is shared in the second license, wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.*

Furthermore, independent claim 42 recites a device for sharing rights adapted to be associated with an item, the device comprising means for receiving *a first license specifying at least one usage right and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices, the least one usage right defines a manner of use selected from a plurality of permitted manners of use for the item, the at least one meta-right defines a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, the at least one meta-right allows the one or more users or devices to transfer rights or to derive new rights, at least one state variable is associated with the at least one right in the first license and identifies a location where a state of rights is tracked, and means for generating in a second license one or more rights based on the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices, at least one state variable is associated with the at least one right that is shared in the second license, and the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.*

Thus, the invention recited in independent claims 40, 41 and 42 includes at least the novel features of specifying in a first license at least one usage right and at least one meta-right for an item, the at least one meta-right allows one or more users or devices to transfer rights or to derive new rights, associating at least one state variable with the at least one right in the first license, wherein the at least one state variable identifies a location where a state of rights is tracked, generating in a second license one or more rights based on the meta-right in the first license, and associating at least one state variable with at least one right that is shared in the second license, wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

By contrast, Downs is directed to a method and apparatus of securely providing data to a user's system, wherein the data is encrypted so as to only be decryptable by a data

decrypting key, the data decrypting key being encrypted using a first public key, and the encrypted data being accessible to the user's system. The method includes transferring the encrypted data decrypting key to a clearing house that possesses a first private key, which corresponds to the first public key; decrypting the data decrypting key using the first private key; re-encrypting the data decrypting key using a second public key; transferring the re-encrypted data decrypting key to the user's system, the user's system possessing a second private key, which corresponds to the second public key; and decrypting the re-encrypted data decrypting key using the second private key. However, Downs fails to disclose, teach or suggest at least the noted features recited in independent claims 40, 41 and 42.

Specifically, and contrary to the Examiner's assertions, Downs fails to disclose meta-rights, as recited in the claims. In particular, meta-rights are defined in the present patent application, as rights to "permit granting of rights to others or the derivation of rights" (see, e.g., Applicants' published Specification, paragraph [0041]):

[0041] **Rights can specify transfer rights, such as distribution rights, and can permit granting of rights to others or the derivation of rights.** Such rights are referred to as "meta-rights". **Meta-rights are the rights that one has to [generate], manipulate, modify, [dispose of] or otherwise derive other meta-rights or usage rights. Meta-rights can be thought of as usage rights to usage rights [or other meta-rights].** Meta-rights can include rights to offer, grant, obtain, transfer, delegate, track, surrender, exchange, and revoke usage rights to/from others. Meta-rights can include the rights to modify any of the conditions associated with other rights. For example, a meta-right may be the right to extend or reduce the scope of a particular right. A meta-right may also be the right to extend or reduce the validation period of a right.

As succinctly stated and defined in the present patent application, usage rights permit actions on an item, such as music, video, digital content, and the like. Such actions can include play, read, view, other uses, and the like, of the item. On the other hand, meta-rights permit actions on rights, such as usage rights and meta-rights. Such actions can include generate, modify, transfer, and the like, of rights.

By contrast, the system of Downs does not disclose meta-rights, but instead merely discloses that content stores can alter usage conditions. For example, and as identified by the Examiner, Downs discloses, at Col. 21, lines 30-42:

The Content Usage Control Layer 505 permits the specification and enforcement of the conditions or restrictions imposed on the use of Content 113 use at the End-User Device(s) 109. The conditions may specify the

number of plays allowed for the Content 113, whether or not a secondary copy of the Content 113 is allowed, the number of secondary copies, and whether or not the Content 113 may be copied to an external portable device. The Content Provider(s) 101 sets the allowable Usage Conditions 517 and transmits them to the Electronic Digital Content Store(s) 103 in a SC (see the License Control Layer 501 section). *The Electronic Digital Content Store(s) 103 can add to or narrow the Usage Conditions 517 as long as it doesn't invalidate the original conditions set by the Content Provider(s) 101.* The Electronic Digital Content Store(s) 103 then transmits all Store Usage Conditions 519 (in a SC) to the End-User Device(s) 109 and the Clearinghouse(s) 105. The Clearinghouse(s) 105 perform Usage Conditions Validation 521 before authorizing the Content 113 release to an End-User Device(s) 109.

Thus, contrary to the Examiner's statements on page 2 of the office action, Col. 2, lines 30-36, of Downs does not disclose the concept of meta-rights as it is recited in the claims and used in this invention. Instead, this portion of Downs merely provides that a content store has the ability to add to or narrow usage conditions, but requires that the usage conditions do not invalidate the original conditions set by the Content Provider. This is distinguishable from meta-rights, as is recited in the claims. Specifically, Downs fails to disclose or suggest a license that includes "at least one usage right and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices," "defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item," and "defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, wherein said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights," as is recited in claim 40, for example.

Accordingly, Applicants again respectfully submit that, while Downs suggests that a store can add to or narrow usage conditions with restrictions, Downs completely fails to disclose or suggest the concept of meta-rights as set forth in the specification and recited in the claims. Accordingly, Downs is silent with respect to the novel meta-rights feature of the invention recited in independent claims 40, 41 and 42.

Moreover, contrary to the Examiner's assertion on page 8 of the Office Action that Downs discloses meta-rights by simply defining "onto what kinds of media the content can be transferred to" (citing Downs, col. 59, lines 52-54), Downs completely fails to disclose,

teach or suggest meta-rights, which allow one or more users or devices to transfer rights or to derive new rights, as recited in independent claims 40, 41 and 42.

In addition, the combination of state variables together with meta-rights further patentably distinguishes the invention recited in independent claims 40, 41 and 42 over Downs. For example, the combination of state variables and meta-rights, advantageously, enables the sharing of rights, wherein one shared right can be derived from another shared right in accordance with a meta-right. In addition, a state variable referring to a location on the device can be used to infer that the right is exclusive to the device, whereas a state variable referring to a location on a server can be used to infer that the right is shared among multiple devices; wherein each device that exercises the right will cause the state variable on the server to be updated.

The following two examples illustrate the use of state variables together with meta-rights in more detail (see, e.g., Applicants' published Specification, paragraphs [0095] and [0099]):

[0095] FIG. 14 is used to illustrate employing of a state variable in deriving inherited usage rights, according to the present invention. In FIG. 14, a derived right can inherit a state variable from meta-rights. For example, a personal computer (PC) of a user, Alice, can be configured to play an e-book according to a license 1403. A personal data assistant (PDA) of Alice also can obtain a right to play the e-book according to offer 1401, if the PC and PDA share the same state variables 1404 and 1405, e.g., "AlicePlayEbook." A derived right 1402 allows Alice also to play the e-book on her PDA as long as the PDA and the PC share a same count limit 1406 of 5 times.

[0099] FIG. 16 is used to illustrate employing of a state variable in deriving rights that are shared among a dynamic set of rights recipients, according to the present invention. In FIG. 16, an offer 1601 specifies that a distributor can issue site licenses to affiliated clubs, allowing 5 members of each club to concurrently view, play, and the like, content, such as an e-book. A corresponding state variable 1607 associated with such a right can be unspecified in the offer 1601. When corresponding rights 1602 and 1603 are issued to affiliated clubs, the corresponding club identities are used to specify state variables 1608 and 1609 in the issued rights. The offers 1602 and 1603 are meta-rights derived from the offer 1601, with offer being assigned the distinct state variables 1608 and 1609. Further rights 1604-1606 can be derived from the offers 1602 and 1603 to be shared among members of each respective club. The licenses 1604 and 1605 are examples of rights derived from the offer 1602, and which inherit the state variable 1608, e.g., "urn:acme:club," whereas the license 1606 inherits the state variable 1609, e.g., "urn:foo:club."

Thus, contrary to the Examiner's statements on pages 2-3 of the office action, a state variable is not simply "the number of copies" or "rental terms." Instead, a state variable references, for example, a counter or variable where "the number of copies" or "rental terms" is maintained, and wherein such a counter or variable can be located on a local device or a remote server. The ability to choose the location of a state keeper instead of a specific number, advantageously, provides a mechanism for the rights owner to control rights sharing.

The invention recited in independent claims 40, 41 and 42 and claims dependent therefrom recognizes and solves the following problems:

[0009] However, there are limitations associated with the above-mentioned paradigms wherein only usage rights and conditions associated with content are specified by content owners or other grantors of rights. Once purchased by an end user, a consumer, or a distributor, of content along with its associated usage rights and conditions has no means to be legally passed on to a next recipient in a distribution chain. Further the associated usage rights have no provision for specifying rights to derive other rights, i.e. Rights to modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, revoke, or the like. Common content distribution models often include a multi-tier distribution and usage chain. Known DRM systems do not facilitate the ability to prescribe rights and conditions for all participants along a content distribution and usage chain. Therefore, it is difficult for a content owner to commercially exploit content unless the owner has a relationship with each party in the distribution chain.

The invention recited in independent claims 40, 41 and 42 and claims dependent therefrom provides the following advantages:

[0090] There are multiple ways to specify the scope of state variables, each of which can affect whether the derivative state variables can be shared, how the derivative state variables can be shared, and the like. For example, a state variable can be local, and solely confined to a recipient or can be global, and shared by a predetermined group of recipients. A global state variable can be shared by a group of recipients not determined when derived rights are issued, but to be specified later, perhaps based on certain rules defined in the license or based on other means. A global state variable can be shared between one or more rights suppliers, predetermined recipients, un-specified recipients, and the like. Advantageously, depending on the sharing employed with a given a business model and the rights granted in the meta-rights, state variables can be created at different stages of the value chain.

Thus, for at least the above reasons, Downs fails to disclose, suggest, or render obvious, each and every feature recited in independent claims 40-42. Accordingly, Applicants respectfully request that the rejection of these claims under 35 U.S.C. § 102(b) in

view of Downs. In addition, dependent claims 2-10, 14-22, 25, 27-35, and 43-54 are also allowable over Downs for at least the reasons set forth above, and also on their own merits.

Moreover, Applicants would like to point out that there is no substantive rejection of pending claims 56-57 in this office action. However, in the hopes of expediting prosecution of this application, Applicants submit that these claims are also allowable over Downs for at least the reasons set forth above, and also on their own merits. If a further office is deemed necessary by the Examiner, Applicants request an indication of allowability for these claims as well.

In view of the foregoing, it is submitted that the present application is in condition for allowance and a notice to that effect is respectfully requested. If, however, the Examiner deems that any issue remains after considering this response, the Examiner is invited to contact the undersigned attorney to expedite the prosecution and engage in a joint effort to work out a mutually satisfactory solution.

Except for issue fees payable under 37 C.F.R. § 1.18, the Commissioner is hereby authorized by this paper to charge any additional fees during the entire pendency of this application including fees due under 37 C.F.R. §§ 1.16 and 1.17 which may be required, including any required extension of time fees, or credit any overpayment to Deposit Account No. 19-2380. This paragraph is intended to be a **CONSTRUCTIVE PETITION FOR EXTENSION OF TIME** in accordance with 37 C.F.R. § 1.136(a)(3).

Respectfully submitted,

Date: October 31, 2007

/Stephen M. Hertzler, Reg. No. 58,247/
Stephen M. Hertzler

NIXON PEABODY LLP
Suite 900, 401 9th Street, N.W.
Washington, D.C. 20004-2128
(202) 585-8000

Electronic Acknowledgement Receipt

EFS ID:	2402375
Application Number:	10956070
International Application Number:	
Confirmation Number:	8299
Title of Invention:	System and method for rights offering and granting using shared state variables
First Named Inventor/Applicant Name:	Mai Nguyen
Customer Number:	22204
Filer:	Stephen M. Hertzler
Filer Authorized By:	
Attorney Docket Number:	111325-235000
Receipt Date:	31-OCT-2007
Filing Date:	04-OCT-2004
Time Stamp:	13:08:39
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment was successfully received in RAM	\$ 1860
RAM confirmation Number	9471
Deposit Account	192380

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:
Charge any Additional Fees required under 37 C.F.R. Section 1.16 and 1.17

File Listing:

Document Number	Document Description	File Name	File Size(Bytes) /Message Digest	Multi Part /.zip	Pages (if appl.)
1		235000_2007-10-31_Response_to_Final_Office_Action.pdf	202201 b9505be43cadb81e2bb44b6542c97e7a25d1cd56	yes	19
Multipart Description/PDF files in .zip description					
	Document Description		Start		End
	Amendment After Final		1		1
	Claims		2		9
	Applicant Arguments/Remarks Made in an Amendment		10		19
Warnings:					
Information:					
2	Request for Continued Examination (RCE)	235000_2007-10-31_RCE_Fillable.pdf	775412 0d081462a864961a29e707360ff0d0dddb9424c48	no	3
Warnings:					
Information:					
3	Fee Worksheet (PTO-06)	fee-info.pdf	8329 a310654d01cd4d4be7fb2445c8de95a4b910be543	no	2
Warnings:					
Information:					
Total Files Size (in bytes):			985942		
<p>This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.</p> <p><u>New Applications Under 35 U.S.C. 111</u> If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.</p> <p><u>National Stage of an International Application under 35 U.S.C. 371</u> If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.</p> <p><u>New International Application Filed with the USPTO as a Receiving Office</u> If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.</p>					

Amendments to the Claims:

1. (Cancelled)

2. (Previously Presented) The method of claim 40, wherein the state variable in the first or second license inherits a state thereof for content usage or rights derivation from other generated usage rights and meta-rights.

3. (Previously Presented) The method of claim 40, wherein the state variable in the first or second license shares a state thereof for content usage or rights derivation with other generated usage rights and meta-rights.

4. (Previously Presented) The method of claim 40, wherein the state variable in the first or second license inherits a remaining state for content usage or rights derivation from other generated usage rights and meta-rights.

5. (Previously Presented) The method of claim 40, wherein the state variable in the first or second license is updated upon exercise of a right associated with the state variable.

6. (Previously Presented) The method of claim 40, wherein the state variable in the first or second license represents a collection of states.

7. (Previously Presented) The method of claim 40, further comprising:
generating in a third license one or more rights from at least one of the usage right and the meta-right in the second license,

wherein the one or more rights in the third license includes at least one right that is shared among one or more users or devices;

associating at least one state variable with the at least one right that is shared in the third license,

wherein the at least one state variable that is associated with the third license is based on the at least one state variable that is associated with the second license.

8. (Previously Presented) The method of claim 40, further comprising a plurality of state variables that determine the state of the at least one right that is shared in the first or the second license.

9. (Cancelled)

10. (Previously Presented) The method of claim 40, wherein the state variable in the second license is transferred from the at least one right in the first license and is associated with the right that is shared in the second license.

11-13. (Cancelled)

14. (Previously Presented) The system of claim 41, wherein the state variable in the first or second license inherits a state thereof for content usage or rights derivation from other generated usage rights and meta-rights.

15. (Previously Presented) The system of claim 41, wherein the state variable in the first or second license shares a state thereof for content usage or rights derivation with other generated usage rights and meta-rights.

16. (Previously Presented) The system of claim 41, wherein the state variable in the first or second license inherits a remaining state for content usage or rights derivation from other generated usage rights and meta-rights.

17. (Previously Presented) The system of claim 41, wherein the state variable in the first or second license is updated upon exercise of a right associated with the state variable.

18. (Previously Presented) The system of claim 41, wherein the state variable in the first or second license represents a collection of states.

19. (Previously Presented) The system of claim 41, further comprising:
means for generating in a third license one or more rights from at least one of the usage right and the meta-right in the second license,
wherein the one or more rights in the third license includes at least one right that is shared among one or more users or devices;
means for associating at least one state variable with the at least one right that is shared in the third license,
wherein the at least one state variable that is associated with the third license is based on the at least one state variable that is associated with the second license.

20. (Previously Presented) The system of claim 41, including a plurality of state variables that determine the state of the at least one right that is shared in the first or the second license.

21. (Cancelled)

22. (Previously Presented) The system of claim 41, wherein the state variable in the second license is transferred from the at least one right in the first license and is associated with the right that is shared in the second license.

23-24. (Cancelled)

25. (Previously Presented) The system of claim 41, wherein the system is implemented with one or more hardware and software components.

26. (Cancelled)

27. (Previously Presented) The device of claim 42, wherein the state variable in the first or second license inherits a state thereof for content usage or rights derivation from other generated usage rights and meta-rights.

28. (Previously Presented) The device of claim 42, wherein the state variable in the first or second license shares a state thereof for content usage or rights derivation with other generated usage rights and meta-rights.

29. (Previously Presented) The device of claim 42, wherein the state variable in the first or second license inherits a remaining state for content usage or rights derivation from other generated usage rights and meta-rights.

30. (Previously Presented) The device of claim 42, wherein the state variable in the first or second license is updated upon exercise of a right associated with the state variable.

31. (Previously Presented) The device of claim 42, wherein the state variable in the first or second license represents a collection of states.

32. (Previously Presented) The device of claim 42, wherein a third license includes one or more rights from at least one of the usage right and the meta-right in the second license,

the one or more rights in the third license includes at least one right that is shared among one or more users or devices,

at least one state variable is associated with the at least one right that is shared in the third license, and

the at least one state variable that is associated with the third license is based on the at least one state variable that is associated with the second license.

33. (Previously Presented) The device of claim 42, including a plurality of state variables that determine the state of the at least one right that is shared in the first or the second license.

34. (Cancelled)

35. (Previously Presented) The device of claim 42, wherein the state variable in the second license is transferred from the at least one right in the first license and is associated with the right that is shared in the second license.

36-39. (Cancelled)

40. (Previously Presented) A method for sharing rights adapted to be associated with an item, the method comprising:

specifying in a first license at least one usage right and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices;

defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item;

defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, wherein said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights;

associating at least one state variable with the at least one right in the first license, wherein the at least one state variable identifies a location where a state of rights is tracked;

generating in a second license one or more rights based on the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices; and

associating at least one state variable with the at least one right that is shared in the second license, wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

41. (Previously Presented) A system for sharing rights adapted to be associated with an item, the system comprising:

means for specifying in a first license at least one usage right and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices;

means for defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item;

means for defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, wherein said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights;

means for associating at least one state variable with the at least one right in the first license, wherein the at least one state variable identifies a location where a state of rights is tracked;

means for generating in a second license one or more rights based on the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices; and

means for associating at least one state variable with the at least one right that is shared in the second license, wherein the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

42. (Previously Presented) A device for sharing rights adapted to be associated with an item, the device comprising:

means for receiving a first license specifying at least one usage right and at least one meta-right for the item, wherein the usage right and the meta-right include at least one right that is shared among one or more users or devices, the least one usage right defines a manner of use selected from a plurality of permitted manners of use for the item, the at least one meta-right defines a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights, at least one state variable is associated with the at least one right in the first license and identifies a location where a state of rights is tracked; and

means for generating in a second license one or more rights based on the meta-right in the first license, wherein the one or more rights in the second license includes at least one right that is shared among one or more users or devices, at least one state variable is associated with the at least one right that is shared in the second license, and the at least one state variable that is associated with the second license is based on the at least one state variable that is associated with the first license.

43. (Previously Presented) The method of claim 40, wherein the method is implemented with one or more hardware and software components configured to perform the steps of the method.

44. (Previously Presented) The method of claim 40, wherein the method is implemented with one or more computer readable instructions embedded on a computer readable medium and configured to cause one or more computer processors to perform the steps of the method.

45. (Previously Presented) The device of claim 42, wherein the device is implemented with one or more hardware and software components.

46-48. (Cancelled)

49. (Previously Presented) The method of claim 40, wherein the plurality of permitted manners of use for the item include copy, transfer, loan, play, print, delete, extract, embed, edit, authorize, install, and un-install the item.

50. (Previously Presented) The system of claim 41, wherein the plurality of permitted manners of use for the item include copy, transfer, loan, play, print, delete, extract, embed, edit, authorize, install, and un-install the item.

51. (Previously Presented) The device of claim 42, wherein the plurality of permitted manners of use for the item include copy, transfer, loan, play, print, delete, extract, embed, edit, authorize, install, and un-install the item.

52. (Previously Presented) The method of claim 40, wherein the plurality of permitted manners of rights derivation for the item include issue, modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, and revoke rights for the item.

53. (Previously Presented) The system of claim 41, wherein the plurality of permitted manners of rights derivation for the item include issue, modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, and revoke rights for the item.

54. (Previously Presented) The device of claim 42, wherein the plurality of permitted manners of rights derivation for the item include issue, modify, transfer, offer, grant, obtain, delegate, track, surrender, exchange, transport, exercise, and revoke rights for the item.

55. (Currently Amended) The method of claim 40, further comprising:
generating in a ~~third~~further license one or more rights based on the meta-right in the second license, wherein the one or more rights in the ~~third~~further license includes at least one right that is shared among one or more users or devices; and
associating at least one state variable with the at least one right that is shared in the ~~third~~further license, wherein the at least one state variable that is associated with the ~~third~~further license is based on the at least one state variable that is associated with the second license.

56. (Previously Presented) The method of claim 40, wherein the at least one state variable that is associated with the second license is the same as the at least one state variable that is associated with the first license, if the at least one state variable that is associated with the first license does not identify an unspecified location.

57. (Previously Presented) The method of claim 40, wherein the at least one state variable that is associated with the second license is assigned a new location identification, if the at least one state variable that is associated with the first license identifies an unspecified location.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:)	Confirmation No.: 8299
Mai NGUYEN, <i>et al.</i>)	Group Art Unit: 3621
Serial No. 10/956,070)	Examiner: Evens J. Augustin
Filed: October 4, 2004)	
For: SYSTEM AND METHOD FOR RIGHTS OFFERING AND GRANTING USING SHARED STATE VARIABLES)	Date: October 31, 2007

AMENDMENT AFTER FINAL

MAIL STOP AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In response to the final Office Action mailed May 21, 2007, reconsideration and allowance of the above-identified application is respectfully requested in view of the following amendments and remarks.

Amendments to the Claims begin on page 2 of this paper.

Remarks begin on page 10 of this paper.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD Substitute for Form PTO-875	Application or Docket Number 10/956,070	Filing Date 10/04/2004	<input type="checkbox"/> To be Mailed
---	---	----------------------------------	---------------------------------------

APPLICATION AS FILED – PART I			OTHER THAN SMALL ENTITY				
	(Column 1)	(Column 2)	SMALL ENTITY <input type="checkbox"/>	OR			
FOR	NUMBER FILED	NUMBER EXTRA	RATE (\$)	FEE (\$)	OR	RATE (\$)	FEE (\$)
<input type="checkbox"/> BASIC FEE <small>(37 CFR 1.16(a), (b), or (c))</small>	N/A	N/A	N/A			N/A	
<input type="checkbox"/> SEARCH FEE <small>(37 CFR 1.16(k), (l), or (m))</small>	N/A	N/A	N/A			N/A	
<input type="checkbox"/> EXAMINATION FEE <small>(37 CFR 1.16(o), (p), or (q))</small>	N/A	N/A	N/A			N/A	
TOTAL CLAIMS <small>(37 CFR 1.16(i))</small>	minus 20 =	*	X \$ =		OR	X \$ =	
INDEPENDENT CLAIMS <small>(37 CFR 1.16(h))</small>	minus 3 =	*	X \$ =			X \$ =	
<input type="checkbox"/> APPLICATION SIZE FEE <small>(37 CFR 1.16(s))</small>	If the specification and drawings exceed 100 sheets of paper, the application size fee due is \$250 (\$125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
<input type="checkbox"/> MULTIPLE DEPENDENT CLAIM PRESENT <small>(37 CFR 1.16(j))</small>							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II					OTHER THAN SMALL ENTITY				
	(Column 1)	(Column 2)	(Column 3)		SMALL ENTITY	OR			
AMENDMENT	10/31/2007	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE (\$)	ADDITIONAL FEE (\$)	OR	RATE (\$)	ADDITIONAL FEE (\$)
	Total (37 CFR 1.16(i))	* 40	Minus ** 39	= 1	X \$ =		OR	X \$50=	50
	Independent (37 CFR 1.16(h))	* 3	Minus ***3	= 0	X \$ =		OR	X \$210=	0
	<input type="checkbox"/> Application Size Fee (37 CFR 1.16(s))								
	<input type="checkbox"/> FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))						OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	50

	(Column 1)	(Column 2)	(Column 3)						
AMENDMENT		CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE (\$)	ADDITIONAL FEE (\$)	OR	RATE (\$)	ADDITIONAL FEE (\$)
	Total (37 CFR 1.16(i))	*	Minus **	=	X \$ =		OR	X \$ =	
	Independent (37 CFR 1.16(h))	*	Minus ***	=	X \$ =		OR	X \$ =	
	<input type="checkbox"/> Application Size Fee (37 CFR 1.16(s))								
	<input type="checkbox"/> FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))						OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
 ** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
 *** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
 The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
Carolyn Cofer

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**
 If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
-----------------	-------------	----------------------	---------------------	------------------

10/956,070	10/04/2004	Mai Nguyen	111325-235000	8299
------------	------------	------------	---------------	------

22204 7590 12/13/2007

NIXON PEABODY, LLP
401 9TH STREET, NW
SUITE 900
WASHINGTON, DC 20004-2128

EXAMINER

AUGUSTIN, EVENS J

ART UNIT	PAPER NUMBER
----------	--------------

3621

MAIL DATE	DELIVERY MODE
-----------	---------------

12/13/2007

PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No. 10/956,070	Applicant(s) NGUYEN ET AL.	
	Examiner Evens Augustin	Art Unit 3621	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) Responsive to communication(s) filed on 10/31/2007.
- 2a) This action is **FINAL**.
- 2b) This action is non-final.
- 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) Claim(s) 2-8, 10, 14-20, 22, 25, 27-33, 35, 40-45 and 49-57 is/are pending in the application.
 - 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) Claim(s) _____ is/are allowed.
- 6) Claim(s) 2-8, 10, 14-20, 22, 25, 27-33, 35, 40-45, and 49-57 is/are rejected.
- 7) Claim(s) _____ is/are objected to.
- 8) Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) The specification is objected to by the Examiner.
- 10) The drawing(s) filed on _____ is/are: a) accepted or b) objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
 - a) All b) Some * c) None of:
 - 1. Certified copies of the priority documents have been received.
 - 2. Certified copies of the priority documents have been received in Application No. _____.
 - 3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

Acknowledgements

1. This is in response to the request for continued examination filed on 10/31/2007. Applicant's remarks/arguments have been considered, but have not been found to be persuasive. Claims 2-8, 10, 14-20, 22, 25, 27-33, 35, 40-45, and 49-57 are pending.
- 2.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

A person shall be entitled to a patent unless –

(a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States. . . .

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

4. Claims 2-10, 14-22, 25, 27-35, and 40-54 are rejected under 35 U.S.C. 102(b) as being anticipated by Downs et al (U.S 6226618).
5. As per claims 2-10, 14-22, 25, 27-35, and 40-54, Downs et al. disclose an invention that broadly relates to the field of electronic commerce and more particularly to a system and related tools for the secure delivery and rights management of digital assets, such as print media, films,

games, and music over global communications networks such as the Internet and the World Wide Web. The invention includes the means and devices (hardware and software combination) (columns 53, lines 65-67, column 54, lines 1-3) to accomplish the items below- Claims 25, 38, 39,41-45,48, 51, 54. The invention comprising of the following:

- A. Owners setting/specifying initial usage rights/licensing (**first license**) for content to the distributors (column 21, lines 30-33) ("**specifying in a first license at least one usage right and at least one meta-right for the item**") – *Claims 40-42*
- B. Example of usage rights include (column 59, lines 38-69 ("**specifying in a first license at least one usage right and at least one meta-right for the item**")), ("**defining, via the at least one usage right, a manner of use selected from a plurality of permitted manners of use for the item**")– *Claims 40, 49-54:*

Compressed Version	384 Kbps
Type of user	Private Consumer
Type of Transaction	Purchase or Rental
Number Of copies	1
Rental Terms	14 Days
Transfer on What Media	Mini Disc or Computer

- C. Owners setting initial usage rights/licensing (**first license**) for content to the distributors (column 21, lines 30-33). Those usage rights can be modified by the digital store (column 21, lines 33-39) to create **secondary licensing** or customized licensing (column 10, lines 15-18) to the end user – ("**wherein the usage right and**

the meta-right include at least one right that is shared among one or more users or devices") *Claims 40-42, 10, 22, 35*

- D. Content providers (entity that supplies the content), providing (equivalent to generating) usage conditions (equivalent to usage rights), The content providers also stipulate that the content stores or distributors can add or narrow the original usage rights (meta-rights or rights derived from the initial usage rights) (column 21, lines 30-36) - *Claims 40-42*
- E. Usage rights being copy restrictions, which is manner in which the content can be used (column 9, 32-34, col. 26, lines 10-12). - *Claims 40-42*
- F. Content stores or distributors can add or narrow the original usage rights (sub-rights) (column 21, lines 30-36) - *Claims 40-42*
- G. The system also defines the manner in which the content can be used (**meta-rights**) such as onto what kinds of media the content can be transferred to (column 59, lines 52-54) – *Claims 40-42, 46-48*
- H. State variable such the number of copies a user is allowed to make (column 59, line 50 or rental terms (column 59, lines 55-60) - *Claims 40-42*
- I. State variables can be the number of copies a user is allowed to make - ("**associating at least one state variable with the at least one right** ") (column 59, line 50 or rental terms (column 59, lines 55-60). Content providers and distributors specify the number of plays and local copies allowed for the Content, and whether or not the Content may be recorded to an external portable device (state variable). Downs et al. keep track of the content's copy/play usage and update the copy/play status (column 20, lines 43-50, column 12, lines 11-12). The system also uses watermarks, as state

variable, to ensure that the content is being played in a compliant user device (col. 7, lines 45-55). Inherently the identity or location of where the content is being played or copied has to be established, in order to determine whether or not a user is compliant. - ("**defining, via the at least one meta-right, a manner of rights derivation selected from a plurality of permitted manners of rights derivation for the item, wherein said at least one meta-right allows said one or more users or devices to transfer rights or to derive new rights**") -*Claims 40-42, 55-57*

- J. The secondary licensing such as restrictions on rental time period can not violate the initial time period set by the initial licensing (column 21, line 35) - ("**generating in a second license one or more rights based on the meta-right in the first license**")
Claims 40-42
- K. The state variable is derived from the usage rights (column 59, line 50) or rental terms (column 59, lines 55-60) - *Claims 2-4, 14-16, 27-29*
- L. The system keeps track of the content's copy/play usage and updates the copy/play status (column 20, lines 49-50) – *Claims 5, 17, 30*
- M. A state variable can represent various other states, for example an item that rented can affect the number of copies that can be made or whether or not copies can be made -
Claims 6, 8, 18, 20, 31,33
- N. The system embeds a code on every copy the content, as it is transferred form user device to the next. When the Digital Content is accessed in a compliant End-User Devices, the End-User Player Application reads the watermark to check the use restrictions and updates the watermark as required. If the requested use of the content does not comply with the usage conditions, e.g., the number of copies has been

exhausted, the End-User Device(s) will not perform the request (column 7, lines 40-55) - *Claims 7, 19, 32*

- O. The content does not specify how the initial set of rights and variable are to modified, as long as it does not violate the initial licensing (column 21, line 35) - *Claims 9, 21, 34*

Conclusion

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Evens Augustin whose telephone number is 571-272-6860. The examiner can normally be reached on 10am - 6pm M-F.

7. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Andrew Fischer can be reached on (571) 272-6779.

/Evens J. Augustin/
Evens J. Augustin
December 10, 2007
Art Unit 3621

Index of Claims



Application/Control No.

10/956,070

Examiner

Evens Augustin

Applicant(s)/Patent under Reexamination

NGUYEN ET AL.

Art Unit

3621

√	Rejected
=	Allowed

-	(Through numeral) Cancelled
÷	Restricted

N	Non-Elected
I	Interference

A	Appeal
O	Objected

Claim		Date				
Final	Original	5/1/07				
	1	-				
	2	√				
	3	√				
	4	√				
	5	√				
	6	√				
	7	√				
	8	√				
	9	-				
	10	√				
	11	-				
	12	-				
	13	-				
	14	√				
	15	√				
	16	√				
	17	√				
	18	√				
	19	√				
	20	√				
	21	-				
	22	√				
	23	-				
	24	-				
	25	√				
	26	-				
	27	√				
	28	√				
	29	√				
	30	√				
	31	√				
	32	√				
	33	√				
	34	-				
	35	√				
	36	-				
	37	-				
	38	-				
	39	-				
	40	√				
	41	√				
	42	√				
	43	√				
	44	√				
	45	√				
	46	-				
	47	-				
	48	-				
	49	√				
	50	√				

Claim		Date				
Final	Original	5/1/07				
	51	√				
	52	√				
	53	√				
	54	√				
	55	√				
	56	√				
	57	√				
	58					
	59					
	60					
	61					
	62					
	63					
	64					
	65					
	66					
	67					
	68					
	69					
	70					
	71					
	72					
	73					
	74					
	75					
	76					
	77					
	78					
	79					
	80					
	81					
	82					
	83					
	84					
	85					
	86					
	87					
	88					
	89					
	90					
	91					
	92					
	93					
	94					
	95					
	96					
	97					
	98					
	99					
	100					

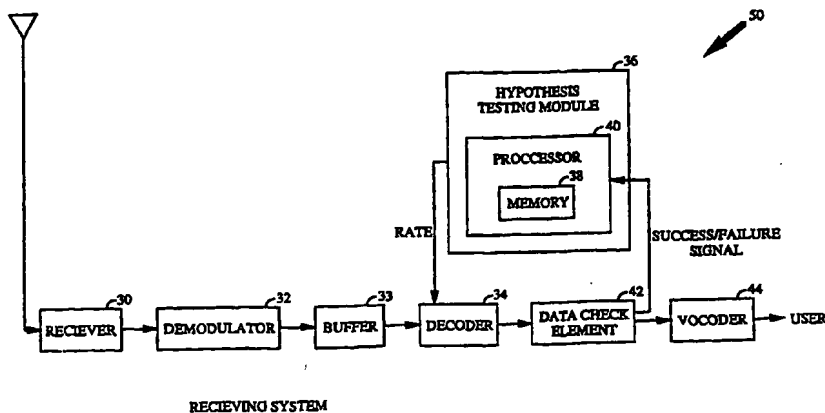
Claim		Date				
Final	Original					
	101					
	102					
	103					
	104					
	105					
	106					
	107					
	108					
	109					
	110					
	111					
	112					
	113					
	114					
	115					
	116					
	117					
	118					
	119					
	120					
	121					
	122					
	123					
	124					
	125					
	126					
	127					
	128					
	129					
	130					
	131					
	132					
	133					
	134					
	135					
	136					
	137					
	138					
	139					
	140					
	141					
	142					
	143					
	144					
	145					
	146					
	147					
	148					
	149					
	150					



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<p>(51) International Patent Classification ⁶ : H04L 25/02</p>	<p>A1</p>	<p>(11) International Publication Number: WO 98/19431 (43) International Publication Date: 7 May 1998 (07.05.98)</p>
<p>(21) International Application Number: PCT/US97/19676 (22) International Filing Date: 27 October 1997 (27.10.97) (30) Priority Data: 08/741,273 30 October 1996 (30.10.96) US (71) Applicant: QUALCOMM INCORPORATED [US/US]; 6455 Lusk Boulevard, San Diego, CA 92121 (US). (72) Inventors: TIEDEMANN, Edward, G., Jr.; 4350 Bromfield Avenue, San Diego, CA 92122 (US). LIN, Yu-Chuan; 585 W. 63rd Avenue, Vancouver, British Columbia V6P 2G7 (CA). (74) Agents: OGROD, Gregory, D. et al.; Qualcomm Incorporated, 6455 Lusk Boulevard, San Diego, CA 92121 (US).</p>		<p>(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).</p> <p>Published <i>With international search report. Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</i></p>

(54) Title: METHOD AND APPARATUS FOR DECODING VARIABLE RATE DATA



(57) Abstract

A system and method for determining the data rate of a frame of data at a receiver (50) of a variable rate communications system. A vocoder at a transmitter encodes a frame of data at one of the rates of a predetermined set of rates. The data rate is dependent on the speech activity during the time frame of the data. The data frame is also formatted with overhead bits, including bits for error detection and detection. At the receiver (50), the data rate for the frame is determined based on hypothesis testing. Because the data rate is based on speech activity, a hypothesis test may be designed based on the statistics of speech activity. The received data frame is first decoded by a decoder (34) into information bits at the most probable rate as provided by the hypothesis testing module (36). Data check element (42) generates error metrics for the decoded information bits. If the error metrics indicate that the information bits are of good quality, then the information bits are presented to a vocoder (44) at the receiver to be processed for interface with the user. If the error metrics indicate that the information bits have not been properly decoded, then decoder (34) decodes the received data frame at the other rates of the set of rates until the actual data rate is determined.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MX	Mexico	UG	Uganda
BY	Belarus	IS	Iceland	MY	Malawi	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

METHOD AND APPARATUS FOR DECODING VARIABLE RATE DATA

BACKGROUND OF THE INVENTION

5

I. Field of the Invention

The present invention relates to digital communications. More particularly, the present invention relates to a novel and improved system and method for determining, at a receiver of a variable rate communication system, the rate at which data has been encoded for transmission.

II. Description of the Related Art

15 The use of code division multiple access (CDMA) modulation techniques is one of several techniques for facilitating communications in which a large number of system users are present. Although other techniques such as time division multiple access (TDMA), frequency division multiple access (FDMA), and AM modulation schemes such as amplitude companded single sideband (ACSSB) are known, CDMA has significant advantages over these other techniques. The use of CDMA techniques in a multiple access communication system is disclosed in U.S. Pat. No. 4,901,307, entitled "SPREAD SPECTRUM MULTIPLE ACCESS COMMUNICATION SYSTEM USING SATELLITE OR TERRESTRIAL REPEATERS," assigned to the assignee of the present invention and incorporated by reference herein.

CDMA systems often employ a variable rate vocoder to encode data so that the data rate can be varied from one data frame to another. An exemplary embodiment of a variable rate vocoder is described in U.S. Pat. No. 5,414,796, entitled "VARIABLE RATE VOCODER," assigned to the assignee of the present invention and incorporated by reference herein. The use of a variable rate communications channel reduces mutual interference by eliminating unnecessary transmissions when there is no useful speech to be transmitted. Algorithms are utilized within the vocoder for generating a varying number of information bits in each frame in accordance with variations in speech activity. For example, a vocoder with a set of four rates may produce 20 millisecond data frames containing 16, 40, 80, or 171 information bits, depending on the activity of the speaker. It is desired to transmit each data frame in a fixed amount of time by varying the transmission rate of communications.

SUBSTITUTE SHEET (RULE 26)

Additional details on the formatting of the vocoder data into data frames are described in U.S. Pat. No. 5,511,073, entitled "METHOD AND APPARATUS FOR THE FORMATTING OF DATA FOR TRANSMISSION," assigned to the assignee of the present invention and herein incorporated by reference. The data frames may be further processed, spread spectrum modulated, and transmitted as described in U.S. Pat. No. 5,103,459, entitled "SYSTEM AND METHOD FOR GENERATING WAVEFORMS IN A CDMA CELLULAR TELEPHONE SYSTEM," assigned to the assignee of the present invention and incorporated by reference herein.

Variable rate systems can be developed which include explicit rate information. If the rate is included as part of a variable rate frame, then the rate is not recoverable until after the frame has already been properly decoded, at which point the rate has already been determined. Rather than including the rate in a variable rate frame, the rate could instead be sent in a non-variable rate portion of the frame. However, only a few bits are typically needed to represent the rate, and these bits cannot be efficiently encoded and interleaved in order to provide error protection for fading communications channels. Furthermore, the rate information is only available after some decoding delay and are subject to error.

Alternatively, variable rate systems can be developed which do not include explicit rate information. One technique for the receiver to determine the rate of a received data frame where the rate information is not explicitly included in the frame is described in copending U.S. Patent Application Serial No. 08/233,570, entitled "METHOD AND APPARATUS FOR DETERMINING DATA RATE OF TRANSMITTED VARIABLE RATE DATA IN A COMMUNICATIONS RECEIVER," filed April 26, 1994, assigned to the assignee of the present invention, and incorporated by reference. Another technique is described in copending U.S. Patent Application Serial No. 08/126,477, entitled "MULTIRATE SERIAL VITERBI DECODER FOR CODE DIVISION MULTIPLE ACCESS SYSTEM APPLICATIONS," filed Sept. 24, 1993, assigned to the assignee of the present invention, and incorporated by reference. According to these techniques, each received data frame is decoded at each of the possible rates. Error metrics, describing the quality of the decoded symbols for each frame decoded at each rate, are provided to a processor. The error metrics may include Cyclic Redundancy Check (CRC) results, Yamamoto Quality Metrics, and Symbol Error Rates. These error metrics are well-known in communications systems. The processor analyzes the error metrics and

SUBSTITUTE SHEET (RULE 26)

determines the most probable rate at which the incoming symbols were transmitted.

Decoding each received data frame at each possible data rate will eventually generate the desired decoded data. However, the search through
5 all possible rates is not the most efficient use of processing resources in a receiver. Also, as higher transmission rates are used, power consumption for determining the transmission rate also increases. This is because there are more bits per frame to be processed. Furthermore, as technology evolves, variable rate systems may utilize larger sets of data rates for
10 communicating information. The use of larger sets of rates will make the exhaustive decoding at all possible rates infeasible. In addition, the decoding delay will not be tolerable for some system applications. Consequently, a more efficient rate determination system is needed in a variable rate communications environment. These problems and deficiencies are clearly
15 felt in the art and are solved by the present invention in the manner described below.

SUMMARY OF THE INVENTION

20 The present invention is a novel and improved system and method for determining the transmission rate of communications in a variable rate communications system. In a variable rate system, the data rate at which a data frame is encoded may be based on the speech activity during the time frame. Because the characteristics of speech are known, probability
25 functions may be defined for the data rates which are dependent on the characteristics of speech. The probability functions may in addition be dependent on the measured statistics of the received data frames. Furthermore, hypothesis tests can be designed based on the probability functions to determine the most likely data rate of a received frame of data.
30 These probability functions may be dependent on the selected service option. For example, the probability functions for data services will be different than for voice services.

At the receiver of the present invention, a processor causes a decoder to decode the received frame of data into information bits at the most
35 probable rate as determined by the hypothesis test. The most probable rate may, for example, be the rate of the previous frame of data. The decoder also generates error metrics for the decoded information bits. The decoded bits and the error metrics are provided to a data check element which checks the decoded bits for correctness. If the error metrics indicate that the

decoded information bits are of good quality, then the information bits are provided to a vocoder which further processes the data and provides speech to the user. Otherwise, a failure signal is presented to the processor. The processor then causes the decoder to decode the received frame of data at
5 other data rates until the correct data rate is found.

BRIEF DESCRIPTION OF THE DRAWINGS

The features, objects, and advantages of the present invention will
10 become more apparent from the detailed description set forth below when taken in conjunction with the drawings in which like reference characters identify correspondingly throughout and wherein:

FIG. 1 is a schematic overview of an exemplary CDMA cellular telephone system;

15 FIG. 2 is a block diagram of a variable rate receiving system with particular reference to the rate determination features of the present invention;

FIGS. 3 and 4 are flow charts illustrating two embodiments of the processing steps involved in rate determination wherein the hypothesis test
20 designates the rate of the previous frame of data as the most probable rate for the current frame of data;

FIGS. 5 and 6 are flow charts illustrating two embodiments of the processing steps involved in rate determination wherein the hypothesis test is based on the a priori probability distribution of the data rates; and

25 FIGS. 7 and 8 are flow charts illustrating two embodiments of the processing steps involved in rate determination wherein the hypothesis test is based on the conditional probability distribution of the data rates.

DETAILED DESCRIPTION OF THE PREFERRED 30 EMBODIMENTS

An exemplary cellular mobile telephone system in which the present invention is embodied is illustrated in FIG. 1. For purposes of example this system is described herein within the context of a CDMA cellular
35 communications system. However, it should be understood that the invention is applicable to other types of communication systems such as personal communication systems (PCS), wireless local loop, private branch exchange (PBX) or other known systems. Furthermore systems utilizing other well known transmission modulation schemes such as TDMA and

FDMA as well as other spread spectrum systems may employ the present invention.

An exemplary cellular system in which the rate determination system of the present invention may be implemented is illustrated in FIG. 1. In FIG. 1, system controller and switch 10 typically include appropriate interface and processing hardware for providing system control information to the cell-sites. Controller 10 controls the routing of telephone calls from the public switched telephone network (PSTN) to the appropriate cell-site for transmission to the appropriate mobile unit. Controller 10 also controls the routing of calls from the mobile units via at least one cell-site to the PSTN. Controller 10 may direct calls between mobile users via the appropriate cell-site stations since such mobile units do not typically communicate directly with one another.

Controller 10 may be coupled to the cell-sites by various means such as dedicated telephone lines, optical fiber links or by radio frequency communications. In FIG. 1, two exemplary cell-sites, 12 and 14, along with two exemplary mobile units, 16 and 18, which include cellular telephones, are illustrated. Arrows 20a-20b and 22a-22b respectively define the possible communication links between cell-site 12 and mobile units 16 and 18. Similarly, arrows 24a-24b and arrows 26a-26b respectively define the possible communication links between cell-site 14 and mobile units 18 and 16.

The cellular system illustrated in FIG. 1 may employ a variable rate data channel for communications between cell-sites 12, 14 and mobile units 16, 18. By example, a vocoder (not shown) may encode sampled voice information into symbols at four different rates according to the IS-95-A standard. The IS-95-A Mobile Station-Base Station Compatibility Standard for Dual Mode Wideband Spread Spectrum Cellular System has been provided by the telecommunications industry association (TIA) for CDMA communications. According to IS-95-A, speech is encoded at approximately 8,550 bits per second (bps), 4,000 bps, 2,000 bps, and 800 bps based on voice activity during a 20 millisecond (ms) frame of data. Each frame of vocoder data is then formatted with overhead bits as 9,600 bps, 4,800 bps, 2,400 bps, and 1,200 bps data frames for transmission. The 9,600 bps frame is referred to as a full rate frame; the 4,800 bps data frame is referred to as a half rate frame; a 2,400 bps data frame is referred to as a quarter rate frame; and a 1,200 bps data frame is referred to as an eighth rate frame. Although this example describes a set of four data rates of the IS-95-A standard, it should be recognized that the present invention is equally applicable in systems

utilizing different transmission rates and/or a different number of variable rates.

By encoding each frame of data based on speech activity, data compression is achievable without impacting the quality of the reconstructed speech. Since speech inherently contains periods of silence, 5 i.e. pauses, the amount of data used to represent these periods can be reduced. Variable rate vocoding most effectively exploits this fact by reducing the data rate for these periods of silence. In a system with a set of four rates as described above, periods of active speech will generally be 10 encoded at full rate, while periods of silence will generally be encoded at eighth rate. Most frames (about 80-90%) are encoded at full or eighth rate. Transitions between active speech and periods of silence will typically be encoded at half or quarter rate. An exemplary encoding technique which compresses data based on speech activity is described in U.S. Pat. No. 15 5,511,073 mentioned above.

The data frames are also formatted with overhead bits, which generally will include additional bits for error correction and detection, such as Cyclic Redundancy Check (CRC) bits. The CRC bits can be used by the decoder to determine whether or not a frame of data has been received 20 correctly. CRC codes are produced by dividing the data block by a predetermined binary polynomial as is described in detail in IS-95-A.

In a preferred embodiment, each frame of symbol data is interleaved by an interleaver, preferably on a bit level basis, to increase time diversity for purposes of error detection. The formatted data frames undergo further 25 processing, which include modulation, frequency upconversion to the radio frequency (RF) and amplification of the signals of data frames, before transmission.

When signals of the variable rate data frames are received by a receiver, the receiver must determine the rate of transmission in order to 30 properly decode the signals. However, the rate of the received frame is not known by the mobile station a priori. Therefore, some other method of ascertaining the rate is necessary.

The present invention accomplishes rate determination through the use of hypothesis testing. Hypothesis tests are designed based on the 35 probability distribution of the data rates of the frames of speech. Although the data rate of each received frame is not known a priori, the probability of receiving a frame at a given rate can be determined. As mentioned above, a variable rate vocoder encodes each frame of speech at one of a set of predetermined rates based on the speech activity during the time frame.

Since the characteristics of speech activity can be modeled, probabilistic functions of the data rates which depend on speech activity can be derived from the model. Hypothesis tests can then be designed based on the probabilistic functions of data rates to determine the most likely data rate for
5 each received frame of data.

The use of hypothesis testing for rate determination in a variable rate receiving system may be better appreciated by referring to FIG. 2. In a CDMA environment, for example, the receiving system 50 of FIG. 2 may be implemented in either a mobile unit or a cell site in order to determine the
10 data rate of received signals. The present invention offers particular advantages because it avoids the exhaustive decoding at all rates. By choosing a hypothesis and checking the hypothesis for correctness, the average amount of processing for each received frame is reduced. This is especially important in the mobile unit because reduced processing, and
15 thereby power consumption, in the decoding process can extend battery life in the receiver.

The variable rate receiving system 50 illustrated in FIG. 2 includes receiver 30 for collecting transmitted signals, including the data signal of interest. Receiver 30 amplifies and frequency downconverts the received
20 signals from the RF frequency band to the intermediate frequency (IF) band.

The IF signals are presented to demodulator 32. The design and implementation of demodulator 32 are described in detail in U.S. Pat. No. 5,490,165, entitled "DEMODULATION ELEMENT ASSIGNMENT IN A SYSTEM CAPABLE OF RECEIVING MULTIPLE SIGNALS," issued Feb. 6,
25 1996, and assigned to the assignee of the present invention, the disclosure of which is incorporated by reference herein. Demodulator 32 demodulates the IF signal to produce a data signal consisting of the symbols of one frame of data. Demodulator 32 generates the data signal by despreading and correlating the IF signal addressed to the receiver. The demodulated data
30 signal is then fed to buffer 33. Buffer 33 stores the demodulated data signal, or the received symbols, until it is properly decoded. Buffer 33 may also be the deinterleaver if the data frame had been interleaved for transmission. Buffer 33 provides the demodulated symbol data to decoder 34.

Hypothesis testing module 36 implements the hypothesis test for
35 determining the data rate of a received frame of data. Hypothesis testing module 36 comprises processor 40, which includes memory 38. The information needed in hypothesis testing such as the decoded rates from the previous frames and the probabilities are stored in memory 38. For each data frame received, processor 40 determines the most probable rate based

on the information stored in memory 38. Processor 40 then presents the most probable data rate to decoder 34 which decodes the data signal at this most probable rate to produce decoded bits.

5 In the exemplary embodiment, decoder 34 is a trellis decoder capable of decoding data of varying rates, such as a Viterbi decoder. The design and implementation of a multirate Viterbi decoder which exhaustively decodes a received signal at all rates of a set of rates is described in the
10 aforementioned U.S. Patent Applications 08/233,570 and 08/126,477. It will be understood by one skilled in the art that the multirate Viterbi decoder may be modified to decode at a selected rate. This may be accomplished by having the Viterbi decoder receive a rate indicator input, in response to which the decoder decodes the data signal according to the rate indicator. Thus, the modified Viterbi decoder may decode a received data frame based on a rate indicator supplied by processor 40 of hypothesis testing module 36.

15 Decoder 34 generates information data bits and error metrics characterizing the information bits. The error metrics include the previously described CRC bits, which were added into the data frames as overhead bits. Decoder 34 may also generate other error metrics, such as the Yamamoto Quality Metric and the Symbol Error Rate (SER). The
20 Yamamoto metric is determined by comparing the differences in the metrics of remerging paths in each step of the Viterbi decoding with a threshold and labeling a path as unreliable if the metric difference is less than a quality threshold. If the final path selected by the Viterbi decoder has been labeled as unreliable at any step, the decoder output is labeled as unreliable.
25 Otherwise, it is labeled as reliable. The Symbol Error Rate is determined by taking the decoded bits, re-encoding these bits to provided re-encoded symbols, and comparing these re-encoded symbols against the received symbols which are stored in buffer 33. The SER is a measure of the mismatching between the re-encoded symbols and the received symbols.
30 The decoded information bits and the error metrics are provided to data check element 42, which determines if the information bits have been correctly decoded.

In a preferred embodiment, data check element 42 first checks the CRC bits. If the CRC check fails, then data check element 42 provides a
35 signal indicative of the failure to processor 40. If the CRC check passes, then data check element 42 determines if the re-encoded SER is below a certain threshold. If the SER is above the threshold, then a signal indicative of failure is provided to processor 40. Otherwise, the data rate provided by hypothesis testing module 36 is determined to be correct, and a success

signal is provided to processor 40, whereupon no further decoding is performed on the data frame. The properly decoded data signal is presented to variable rate vocoder 44.

When processor 40 receives a failure signal indicating that data symbols have not been properly decoded into information bits, processor 40 will determine at least one other data rate from the set of data rates at which to decode the data symbols. Processor 40 provides the rate information to decoder 34, which decodes the data symbols at the rate provided. For each data rate at which the data signal is decoded, data check element 42 will determine the quality of the decoded information bits. Upon determination by data check element 42 that the correct data rate has been found, a signal of decoded information bits is provided to variable rate vocoder 44. Vocoder 44 will then process the information bits for interface with the user.

Hypothesis testing module 36 may implement any of a number of hypothesis tests for determining the data rate of a received frame of data. For example the hypothesis test may be based on known statistics of speech activity. It is known that for a set of four rates using 20 ms frames, a full rate frame will usually be followed by another full rate frame, while an eighth-rate frame will usually be followed by another eighth rate frame. Further, it is also known that most frames will either be full or eighth rate rather than half or quarter rate, because the periods of speech and silence do not occur in 20 ms bursts. Based on these characteristics, the hypothesis test may designate the rate of the previous frame of data as the most probable rate for the currently received frame of data.

In an exemplary implementation, the rate of the previous frame of data is stored in memory 38 of hypothesis testing module 36. When a data frame is received, processor 40 of hypothesis testing module 36 obtains the rate of the previous frame from memory 38 and presents it to decoder 34. Decoder 34 decodes the received data frame at the rate of the previous frame to produce information bits. Decoder 34 also generates error metrics which are then presented to data check element 42 along with the information bits. If data check element 42 determines from the error metrics that the decoded bits are of good quality, then the information bits are presented to vocoder 44. Otherwise, a failure indication is sent from data check element 42 to processor 40. Processor 40 may then have decoder 34 exhaustively decode the data frame at all other rates before determining the data rate. A flow chart illustrating some of the steps involved in rate determination as described in the embodiment above is shown in FIG. 3.

Alternatively, processor 40 may have decoder 34 sequentially decode the data frame according to a ranking from the next most likely rate to the least likely rate. The ranking may be determined in a number of ways, such as according to the probability distributions described below. For each
5 decoding, error metrics are generated by decoder 34 and checked by data check element 42 for correctness. When correctly decoded, the decoded frame is passed on to vocoder 44. A flow chart illustrating some of the processing steps of this embodiment is shown in FIG. 4.

Another implementation of hypothesis testing module 36 is based
10 upon the a priori probability distribution of data rates. For a set of four rates, the a priori probability distribution (P) of the data rates may be defined as:

$$P = \text{Prob}\{R_t\}, \quad (3)$$

15 where R_t refers to the full, half, quarter, or eighth rate at time t . The likelihood of receiving a frame at each of the different data rates of a set of rates are maintained in memory 38 of processor 40. Generally, the probability distribution of the data rates are determined based on the theoretical statistics or the empirical statistics of speech activity. The
20 likelihood of receiving a frame at the different rates are then permanently stored in memory 38 for determining the rate of every received frame of data. In a more sophisticated embodiment, the likelihood of the rates stored in memory 38 may be updated based on the actual statistics of the received frames of data.

25 For each new frame of data received, processor 40 obtains the most probable rate from memory 38 and presents the most probable rate to decoder 34. Decoder 34 decodes the data signal at this most probable data rate and presents the decoded data to data check element 42. Error metrics, including the CRC, are also generated by decoder 34 and presented to data
30 check element 42. Other error metrics may also be generated for checking by data check element 42. If the error metrics indicate that the decoded bits are of good quality, then the information bits are presented to vocoder 44. Otherwise, a failure indication is sent from data check element 42 to processor 40. Then, processor 40 obtains the second most likely data rate
35 from memory 38 and presents it to decoder 34, and the process of decoding and error checking is continued until the correct data rate is found. A flow chart of the processing steps of this embodiment is illustrated in FIG. 5. Alternatively, upon receipt of a failure signal by processor 40, processor 40 may cause decoder 34 to exhaustively decode the data frame at each of the

other data rates of the set of rates, and error metrics are checked for each decoding in order to determine the actual rate of transmission. A flow chart of the processing steps of this embodiment is illustrated in FIG. 6.

5 Instead of designing the hypothesis test based on the simple probability distribution of the data rates, conditional probabilities may be used to improve on the accuracy of the rate determination. For example, the probability of receiving a data frame at a given rate may be defined to be conditioned on the actual rates of the previous frames of data. Conditional probabilities based on the previous rates work well because transition
10 characteristics of the data signals are well known. For example, if the rate two frames ago was eighth rate and the rate for the previous frame was half rate, then the most likely rate for the current frame is full rate, because the transition to half rate indicates the onset of active speech. Conversely, if the rate two frames ago was full rate and the rate for the previous frame was
15 quarter rate, then the most likely rate for the present frame might be eighth rate, because the rate transition indicates the onset of silence.

The probability distribution of the data rates conditioned on the rates of the previous n frames of data may be defined as:

$$20 \quad P = \text{Prob}\{ R_t \mid R_{t-1}, R_{t-2}, \dots, R_{t-n} \} \quad (4)$$

where R_t again refers to the rate at time t, and $R_{t-1}, R_{t-2}, \dots, R_{t-n}$ refers to rate(s) of the previous n frame(s) of data, for $n \geq 1$. The likelihood of receiving a frame at each of the different data rates of a set of rates
25 conditioned on the previous n actual rates are stored in memory 38 of processor 40. In addition, the actual data rates of the previous n frames of data are maintained by processor 40, and may be stored in memory 38 as the rates are determined.

For each received frame of data, processor 40 will determine the most
30 probable data rate conditioned on the previous n actual data rates and present it to decoder 34. Decoder 34 will decode the frame at this most probable data rate and present the decoded bits to data check element 42. In addition, error metrics are generated by decoder 34 and presented to data check element 42. If the error metrics indicate that the decoded bits are of
35 good quality, then the information bits are presented to vocoder 44. Also, processor 40 is informed of the rate decision so that it can maintain the history of chosen rates. That is, processor 40 is supplied R_t so that it can be used in determining $\text{Prob}\{ R_t \mid R_{t-1}, R_{t-2}, \dots, R_{t-n} \}$ for the next frame. If error metrics indicate an unsuccessful decoding, then a failure indication signal is

sent from data check element 42 to processor 40, and processor 40 determines the second most probable data rate conditioned on the previous n actual data rates to decode the data frame. As in the simple probabilities case, the process of decoding and error checking is continued until the correct data rate is found. Some of the processing steps of this embodiment are illustrated in a flow chart in FIG. 7. Also as in the simple probabilities case, after a failed decoding at the most likely rate, decoder 34 may exhaustively decode the data frame at all of the other data rates and have error metrics checked for all decoding in order to determine the data rate. Some of the processing steps of this embodiment are illustrated in a flow chart in FIG. 8.

It should be understood that the conditional probability distribution of the data rates may depend on statistics other than the actual rates of the previous frames of data. For example, the probability distribution may be conditioned on one or more frame quality measurements. The probability distribution is then defined to be:

$$P = \text{Prob}\{R_t \mid X_1, X_2, \dots, X_k\}, \quad (5)$$

where R_t is the rate at time t , and X_1, X_2, \dots, X_k are one or more frame quality measurements. The k frame quality measurements may be measurements performed on the current frame of data, or measurements performed on previous frame(s) of data, or a combination of both. An example of a frame quality measurement is the SER error metric mentioned above. Thus, the probability of receiving a frame at a given rate is conditioned on the SER obtained from the previous decoding if a previous decoding had been performed.

The conditional probability distribution may also depend on a combination of the actual rates of the previous frames of data and the frame quality measurements. In this case, the probability distribution of the data rates is defined as:

$$P_t = \text{Prob}\{R_t \mid R_{t-1}, R_{t-2}, \dots, R_{t-n}, X_1, X_2, \dots, X_k\}, \quad (6)$$

where R_t is the rate at time t , $R_{t-1}, R_{t-2}, \dots, R_{t-n}$ are the rates of the previous frames of data and X_1, X_2, \dots, X_k are the frame quality measurements.

In the cases where the probability distribution is based on frame quality measurements, the frame quality measurements should be maintained in processor 40 of hypothesis testing module 36. As can be seen from the above description, the hypothesized frame rate may be conditioned

on a number of different statistics, and the rates of the previous frames and the frame quality measurements are examples of these statistics. For each data frame received, processor 40 uses the statistics to determine the rate at which to decode the frame.

5 A further refinement to the determination of the rate at which to decode a received frame of data considers the processing costs of decoding the frame at the various rates in conjunction with hypothesis testing. In this embodiment, an optimum test sequence of the rates is established based on both the probability distribution of the data rates and the cost of decoding
 10 at each of the data rates. The optimum test sequence is maintained by processor 40, which causes decoder 34 to sequentially decode a received frame of data according to the optimum sequence until the correct rate is found. The optimum test sequence is established to minimize the total expected cost of the rate search. Denoting P_i to be the probability that the rate search will stop at test T_i , and C_i to be the cost for conducting test T_i , the total expected cost of the rate search using test sequence T_1, T_2, \dots, T_M , where M is the number of possible rates in the system and $1 \leq i \leq M$, can be modeled as:

$$20 \quad C_{\text{total}} = C_1 * P_1 + (C_1 + C_2) * P_2 + \dots (C_1 + C_2 + \dots + C_M) * P_M. \quad (7)$$

The optimum test sequence is found by minimizing the total expected cost C_{total} .

25 In Equation (7), the cost C_i for conducting test T_i will generally be the processing power required for decoding a frame at the rate specified by test T_i . The cost may be assigned to be proportional to the frame rate specified by the test T_i because the computational complexity of decoder 34 is in general approximately proportional to the number of bits per frame. The
 30 probabilities P_i may be assigned by the unconditioned a priori probability distribution of data rates as defined by Equation (3), or any of the conditional probability distributions defined by Equations (4), (5), or (6) above.

In a variable rate communications system where data frames are transmitted at 9,600 bps, 4,800 bps, 2,400 bps, and 1,200 bps, the following
 35 example illustrates the formulation of the optimum test sequence for rate determination of a received frame. The costs to decode the 9,600 bps, 4,800 bps, 2,400 bps, and 1,200 bps frames are assumed to be 9.6, 4.8, 2.4, and 1.2, respectively. Further, the probability of receiving a frame at each of the four

rates is assumed to be the unconditioned a priori probabilities having the following values:

$$\text{Prob}(9,600 \text{ bps}) = 0.291, \quad (8)$$

$$5 \quad \text{Prob}(4,800 \text{ bps}) = 0.039, \quad (9)$$

$$\text{Prob}(2,400 \text{ bps}) = 0.072, \text{ and} \quad (10)$$

$$\text{Prob}(1,200 \text{ bps}) = 0.598. \quad (11)$$

10 The probabilities given in Equations (8)-(11) are derived from steady state empirical data.

A listing of all possible test sequences for rate determination in the system transmitting frames at 9,600, 4,800, 2,400, and 1,200 bps is shown in Table I below. In Table I, column 1 lists all possible test sequences T_1, T_2, T_3, T_4 , where $T_i = 1$ refers to the test of decoding at 9,600 bps, $T_i = 1/2$ refers to the test of decoding at 4,800 bps, $T_i = 1/4$ refers to the test of decoding at 2,400 bps, and $T_i = 1/8$ refers to the test of decoding at 1,200 bps. Columns 2 and 3 list the probability P_1 and the cost C_1 of performing the test T_1 , columns 4 and 5 list the probability P_2 and the cost C_2 of performing the test T_2 , columns 6 and 7 list the probability P_3 and the cost C_3 of performing the test T_3 , and columns 8 and 9 list the probability P_4 and the cost C_4 of performing the test T_4 . The total cost C_{total} of performing the test sequence T_1, T_2, T_3, T_4 is listed in column 10.

T ₁ , T ₂ , T ₃ , T ₄	P ₁	C ₁	P ₂	C ₂	P ₃	C ₃	P ₄	C ₄	C _{total}
1, 1/2, 1/4, 1/8	0.291	9.6	0.039	4.8	0.072	2.4	0.598	1.2	15.33
1, 1/2, 1/8, 1/4	0.291	9.6	0.039	4.8	0.598	1.2	0.072	2.4	13.98
1, 1/4, 1/2, 1/8	0.291	9.6	0.072	2.4	0.039	4.8	0.598	1.2	15.08
1, 1/4, 1/8, 1/2	0.291	9.6	0.072	2.4	0.598	1.2	0.039	4.8	12.25
1, 1/8, 1/2, 1/4	0.291	9.6	0.598	1.2	0.039	4.8	0.072	2.4	11.16
1, 1/8, 1/4, 1/2	0.291	9.6	0.598	1.2	0.072	2.4	0.039	4.8	10.90
1/2, 1, 1/4, 1/8	0.039	4.8	0.291	9.6	0.072	2.4	0.598	1.2	16.35
1/2, 1, 1/8, 1/4	0.039	4.8	0.291	9.6	0.598	1.2	0.072	2.4	15.00
1/2, 1/4, 1, 1/8	0.039	4.8	0.072	2.4	0.291	9.6	0.598	1.2	16.36
1/2, 1/4, 1/8, 1	0.039	4.8	0.072	2.4	0.598	1.2	0.291	9.6	10.97
1/2, 1/8, 1, 1/4	0.039	4.8	0.598	1.2	0.291	9.6	0.072	2.4	9.61
1/2, 1/8, 1/4, 1	0.039	4.8	0.598	1.2	0.072	2.4	0.291	9.6	9.62
1/4, 1, 1/2, 1/8	0.072	2.4	0.291	9.6	0.039	4.8	0.598	1.2	15.08
1/4, 1, 1/8, 1/2	0.072	2.4	0.291	9.6	0.598	1.2	0.039	4.8	12.26
1/4, 1/2, 1, 1/8	0.072	2.4	0.039	4.8	0.291	9.6	0.598	1.2	16.11
1/4, 1/2, 1/8, 1	0.072	2.4	0.039	4.8	0.598	1.2	0.291	9.6	10.71
1/4, 1/8, 1/2, 1	0.072	2.4	0.598	1.2	0.039	4.8	0.291	9.6	7.89
1/4, 1/8, 1, 1/2	0.072	2.4	0.598	1.2	0.291	9.6	0.039	4.8	6.87
1/8, 1, 1/4, 1/2	0.598	1.2	0.291	9.6	0.072	2.4	0.039	4.8	5.51
1/8, 1, 1/2, 1/4	0.598	1.2	0.291	9.6	0.039	4.8	0.072	2.4	5.76
1/8, 1/2, 1, 1/4	0.598	1.2	0.039	4.8	0.291	9.6	0.072	2.4	6.79
1/8, 1/2, 1/4, 1	0.598	1.2	0.039	4.8	0.072	2.4	0.291	9.6	6.79
1/8, 1/4, 1/2, 1	0.598	1.2	0.072	2.4	0.039	4.8	0.291	9.6	6.54
1/8, 1/4, 1, 1/2	0.598	1.2	0.072	2.4	0.291	9.6	0.039	4.8	5.52

Table I

5 As shown in Table I, the optimum test sequence is the sequence 1/8,
 1, 1/4, 1/2 shown in the 19th row. This test sequence offers the lowest total
 expected cost of processing. Therefore, the rate determination system would
 decode a received frame of data at 1,200 bps first. If the decoding at 1,200 bps
 is not successful, then the frame would be decoded sequentially at 9,600 bps,
 10 2,400 bps, and 4,800 bps until the correct rate is found. In a preferred
 embodiment, the optimum test sequence is maintained by processor 40 of
 hypothesis testing module 36. For each frame of data received, processor 40
 causes decoder 34 to decode the frame sequentially according to the
 optimum test sequence, with each decoding checked by data check element
 15 42, until the correct data rate is found. Processing resources are efficiently
 utilized in this rate determination system because the decoding is performed
 sequentially according to an optimum search sequence.

Based on the embodiments described above, it will be understood by
 one skilled in the art that the present invention is applicable to all systems

in which data has been encoded according to a variable rate scheme and the data must be decoded in order to determine the rate. Even more generally, the invention is applicable to all systems in which the encoded data E is a function of the data D and some key k, and there exists some information in
5 D or E which permits the verification of the correct D by the receiver. The sequence k may be time varying. The encoded data is represented as:

$$E = f(D,k), \quad (1)$$

10 where k is from a small set K of keys and where some probability function exists on the set of keys. The inverse of the encoding, or the decoding, can be represented as:

$$D = f^{-1}(E,k), \quad (2)$$

15

where k is chosen so that D is correct.

As an example, assume that D is data composed of fixed-length sequence D1 and fixed length sequence D2 so that $D = D1, D2$. Sequence D2 is the Cyclic Redundancy Code (CRC) of D1, so that $D2 = \text{frcr}(D1)$. Assume also
20 that the encoding function, $f(D,k)$, is an exclusive-OR of a fixed-length D with the fixed length sequence k. Then, the decoding, $f^{-1}(E,k)$, would be the exclusive-OR of E with the correct k. The correct k is verified by checking whether $D2 = \text{frcr}(D1)$. The correct k can be found by decoding all possible k's in K and then determining whether the CRC check passes.
25 Alternatively, it can be done by sequentially decoding using one k at a time, with no further decoding once the "correct" k is found. According to the present invention, the order of sequential decoding is to be determined by hypothesis testing. A number of hypothesis tests, including the tests described above, may be utilized. The order of sequential decoding may in
30 addition depend on the cost of processing, as described above. The use of hypothesis testing and/or cost functions in formulating a test sequence for rate determination reduces the average amount of processing as fewer k's will have to be tried.

The previous description of the preferred embodiments is provided
35 to enable any person skilled in the art to make or use the present invention. The various modifications to these embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments without the use of the inventive faculty. Thus, the present invention is not intended to be limited to the

embodiments shown herein but is to be accorded the widest scope consistent with the principles and novel features disclosed herein.

CLAIMS

1. In a variable rate communications system, a sub-system for
2 determining, at a receiver, the data rate of a received data frame, comprising:
a processor for generating a signal indicating the most likely rate of
4 said received data frame in accordance with a predetermined hypothesis test;
and
6 a decoder for receiving said most likely rate signal and for decoding
said received data frame into a decoded frame of bits at said most likely rate.
2. The rate determination sub-system of claim 1 wherein said
2 most likely rate is the rate of the previous data frame.
3. The rate determination sub-system of claim 1 wherein said
2 hypothesis test is based on an a priori probability distribution of data rates.
4. The rate determination sub-system of claim 1 wherein said
2 hypothesis test is based on a conditional probability distribution of data rates
conditioned on the rate of at least one previous data frame.
5. The rate determination sub-system of claim 1 wherein said
2 hypothesis test is based on a conditional probability distribution of data rates
conditioned on at least one frame quality measurement.
6. The rate determination sub-system of claim 1 further
2 comprising a data check element for receiving said decoded bits, generating
error metrics characterizing said decoded bits, and generating a quality
4 indication based on said error metrics for said decoded bits.
7. The rate determination sub-system of claim 6,
2 further comprising a vocoder for receiving said decoded bits and
processing said decoded bits to provide speech to an user upon generation of
4 a positive indication of said quality; and
wherein upon generation of a negative indication of said quality, said
6 processor further causes said decoder to perform additional decoding of said
received data frame in accordance with at least one rate other than said most
8 likely rate.

8. The rate determination sub-system of claim 7,
2 wherein said additional decoding is performed sequentially in
accordance with a predetermined test sequence of data rates;
4 wherein said data check element generates error metrics for each said
additional decoding and generates a quality indication based on said error
6 metrics for each said additional decoding; and
8 wherein said additional decoding terminates upon generation of a
positive indication of said quality.

9. The rate determination sub-system of claim 7,
2 wherein said additional decoding comprises exhaustive decoding of
said received data frame at all rates of a rate set except said most likely rate;
4 and
wherein said data check element generates error metrics for each said
6 additional decoding and determines the rate of said received data frame in
accordance with said error metrics.

10. The rate determination sub-system of claim 6 wherein said
2 error metrics include a Cyclic Redundancy Check result.

11. The rate determination sub-system of claim 6 wherein said
2 error metrics include a Symbol Error Rate metric.

12. The rate determination sub-system of claim 6 wherein said
2 error metrics include a Yamamoto quality metric.

13. The rate determination sub-system of claim 1 wherein said
2 processor comprises a memory for storing said most likely rate.

14. The rate determination sub-system of claim 1 wherein said
2 decoder is a Viterbi decoder.

15. In a variable rate communications system, a sub-system for
2 determining, at a receiver, the data rate of a received data frame, comprising:
a processor for generating a test sequence of data rates for determining
4 the rate of a received data frame, said test sequence being generated in
accordance with a predetermined hypothesis test;

6 a decoder for decoding said received data frame sequentially according
to said test sequence and generating a decoded frame of bits for each rate at
8 which said received data frame is decoded;

a data check element for generating error metrics characterizing said
10 decoded bits and for generating a quality indication based on said error
metrics for each rate at which said received data frame is decoded; and

12 wherein no further decoding is performed upon generation of a
positive indication of said quality.

16. The rate determination sub-system of claim 15 wherein said
2 hypothesis test is based on an a priori probability distribution of data rates.

17. The rate determination sub-system of claim 15 wherein said
2 hypothesis test is based on a conditional probability distribution of data rates
conditioned on the rate of at least one previous data frame.

18. The rate determination sub-system of claim 15 wherein said
2 hypothesis test is based on a conditional probability distribution of data rates
conditioned on at least one frame quality measurement.

19. The rate determination sub-system of claim 16 wherein said
2 test sequence is generated further in accordance with the cost of decoding
said received data frame at each of said data rates.

20. The rate determination sub-system of claim 17 wherein said
2 test sequence is generated further in accordance with the cost of decoding
said received data frame at each of said data rates.

21. The rate determination sub-system of claim 18 wherein said
2 test sequence is generated further in accordance with the cost of decoding
said received data frame at each of said data rates.

22. The rate determination sub-system of claim 15 further
2 comprising a vocoder for receiving said decoded bits and processing said
decoded bits to provide speech to an user upon generation of a positive
4 indication of said quality.

23. The rate determination sub-system of claim 15 wherein said
2 error metrics include a Cyclic Redundancy Check result.

24. The rate determination sub-system of claim 15 wherein said
2 error metrics include a Symbol Error Rate metric.

25. The rate determination sub-system of claim 15 wherein said
2 error metrics include a Yamamoto quality metric.

26. The rate determination sub-system of claim 15 wherein said
2 processor comprises a memory for storing said test sequence of data rates.

27. The rate determination sub-system of claim 15 wherein said
2 decoder is a Viterbi decoder.

28. A method for determining the rate of a received data frame in
2 a variable rate communications system, comprising the steps of:
receiving a wide-band signal;
4 demodulating said wide-band signal to produce a data signal, wherein
said data signal has been transmitted at one of a set of possible transmission
6 rates;
generating a test sequence of data rates for determining the rate of said
8 data signal, said test sequence being generated in accordance with a
predetermined hypothesis test;
10 decoding said data signal sequentially according to said test sequence
to generate a decoded frame of bits for each rate at which said data signal is
12 decoded;
generating error metrics characterizing said decoded frame of bits for
14 each rate at which said data signal is decoded;
generating a quality indication based on said error metrics for each
16 rate at which said data signal is decoded; and
upon generation of a positive indication of said quality, providing
18 said decoded frame of bits to a vocoder which processes said decoded bits to
provide speech to an user.

29. The method of claim 28 wherein said hypothesis test is based
2 on an a priori probability distribution of data rates.

30. The method of claim 28 wherein said hypothesis test is based
2 on a conditional probability distribution of data rates conditioned on the rate
of at least one previous data frame.

31. The method of claim 28 wherein said hypothesis test is based
2 on a conditional probability distribution of data rates conditioned on at least
one frame quality measurement.

32. The method of claim 29 wherein said test sequence is generated
2 further in accordance with the cost of decoding said received data frame at
each of said data rates.

33. The method of claim 30 wherein said test sequence is generated
2 further in accordance with the cost of decoding said received data frame at
each of said data rates.

34. The method of claim 31 wherein said test sequence is generated
2 further in accordance with the cost of decoding said received data frame at
each of said data rates.

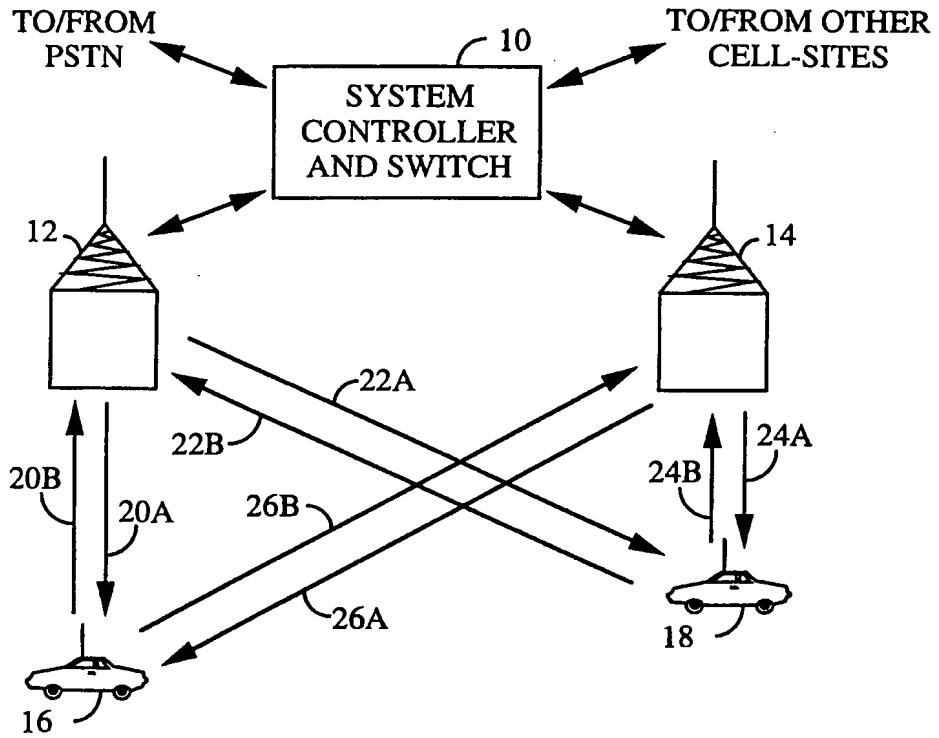


FIG. 1

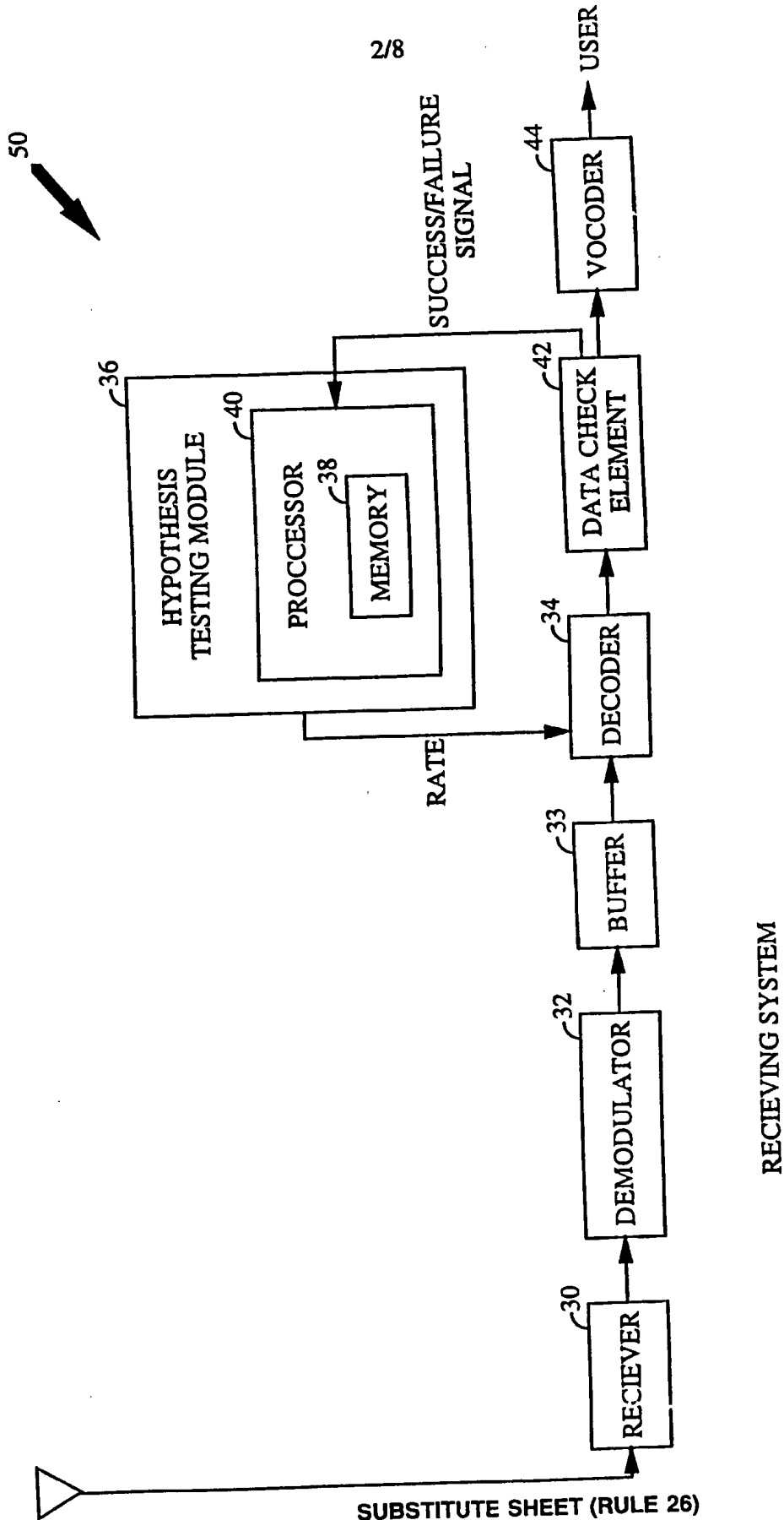


FIG. 2

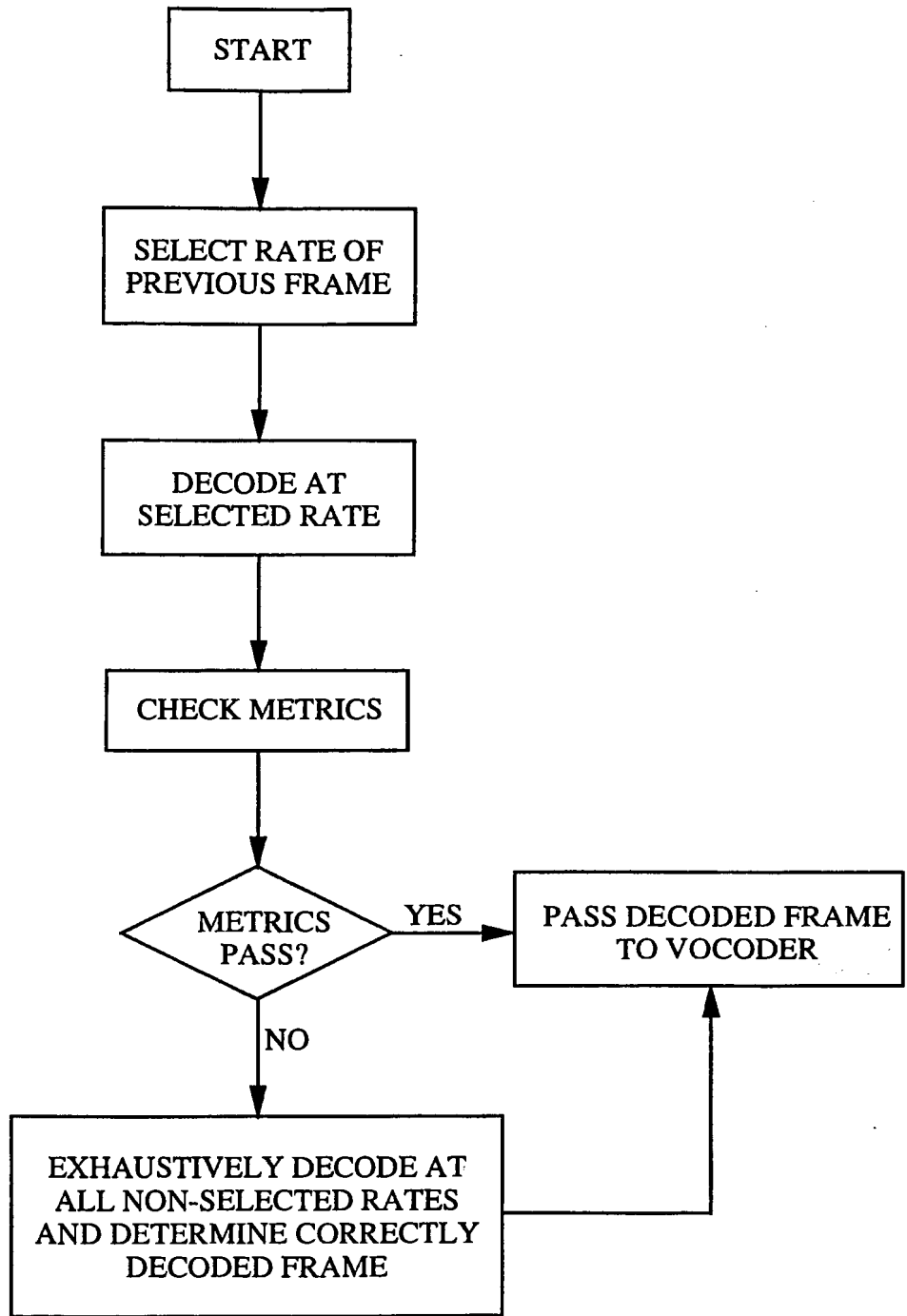


FIG. 3

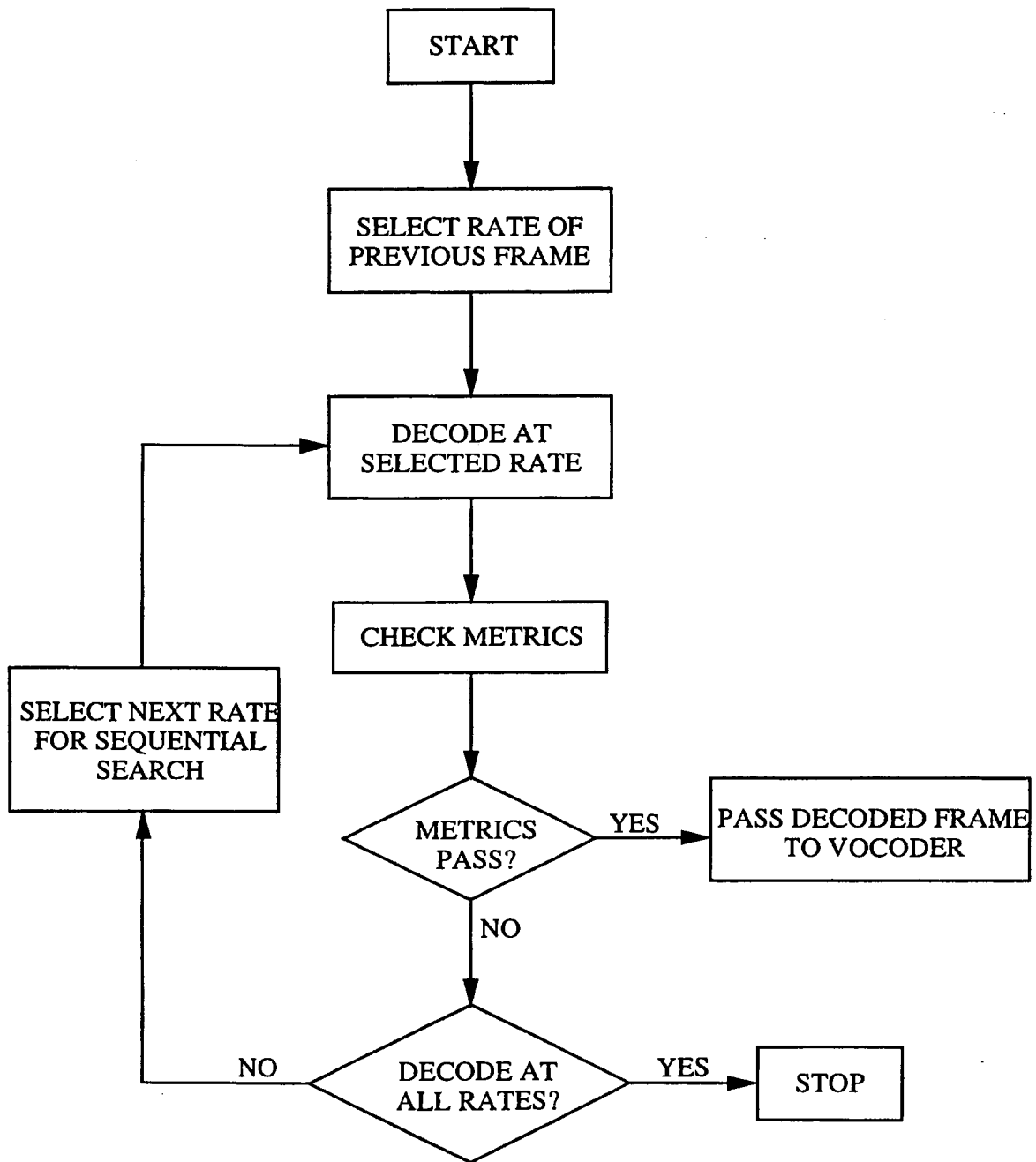


FIG. 4

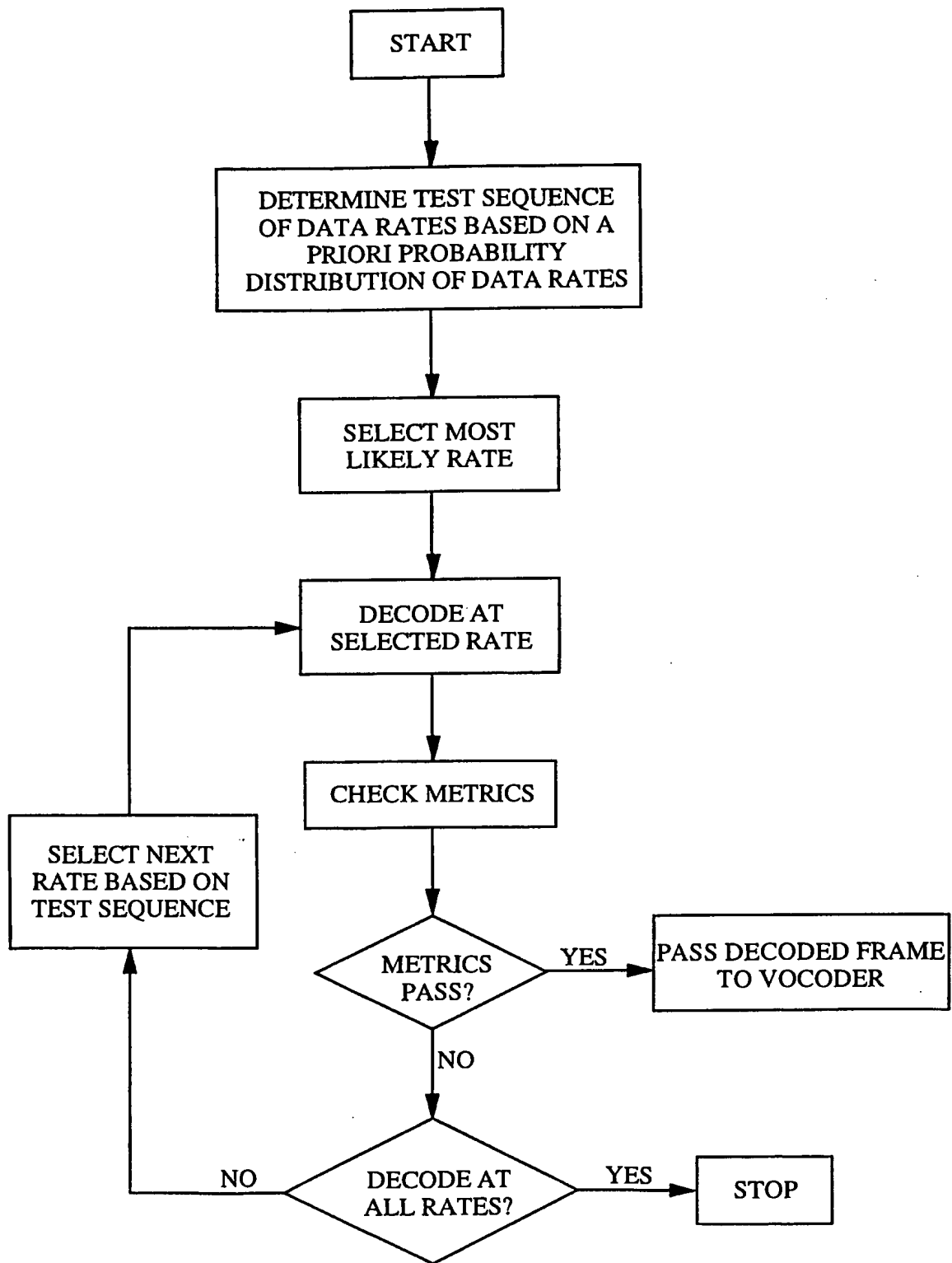


FIG. 5

6/8

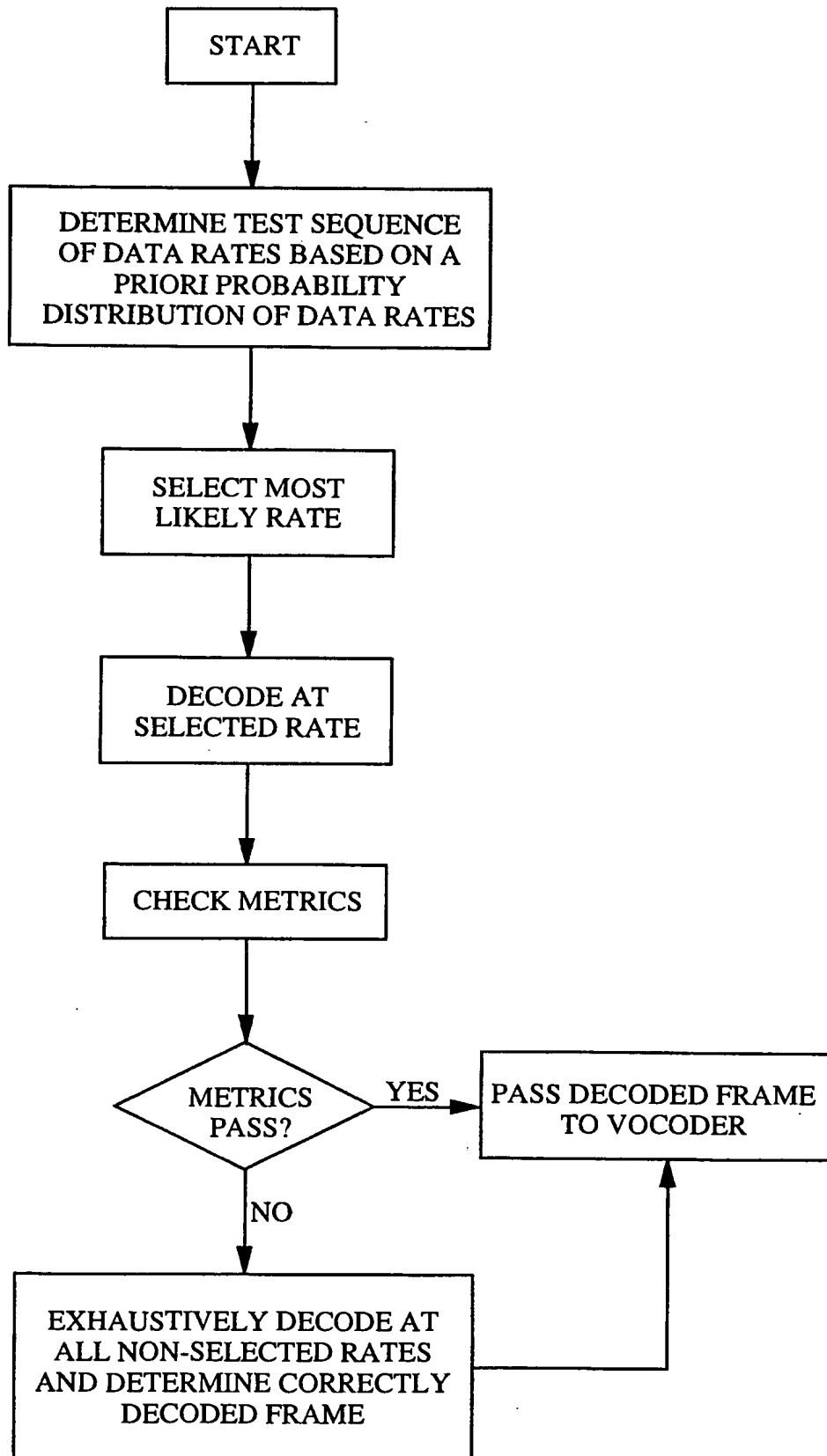


FIG. 6

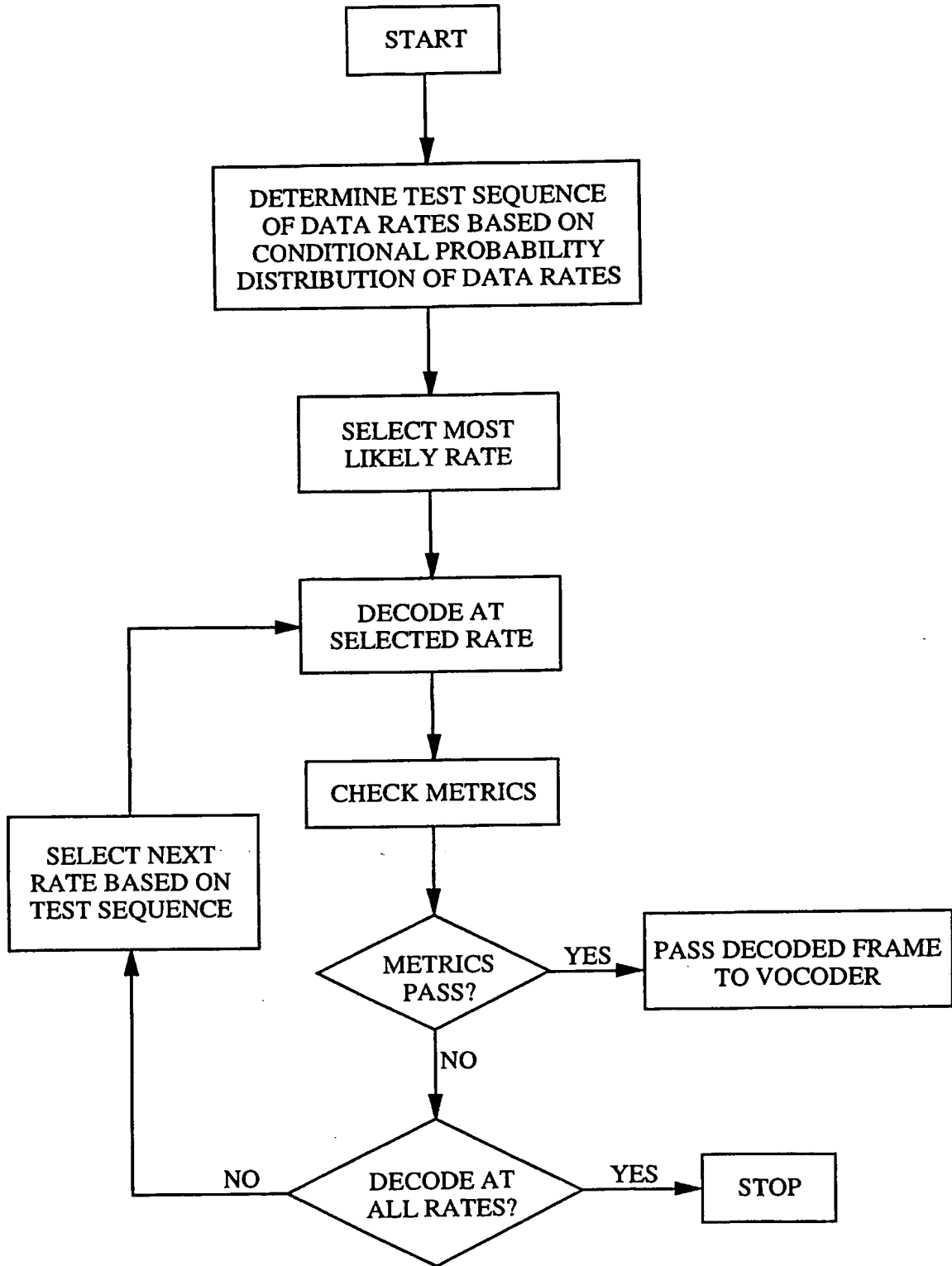


FIG. 7

8/8

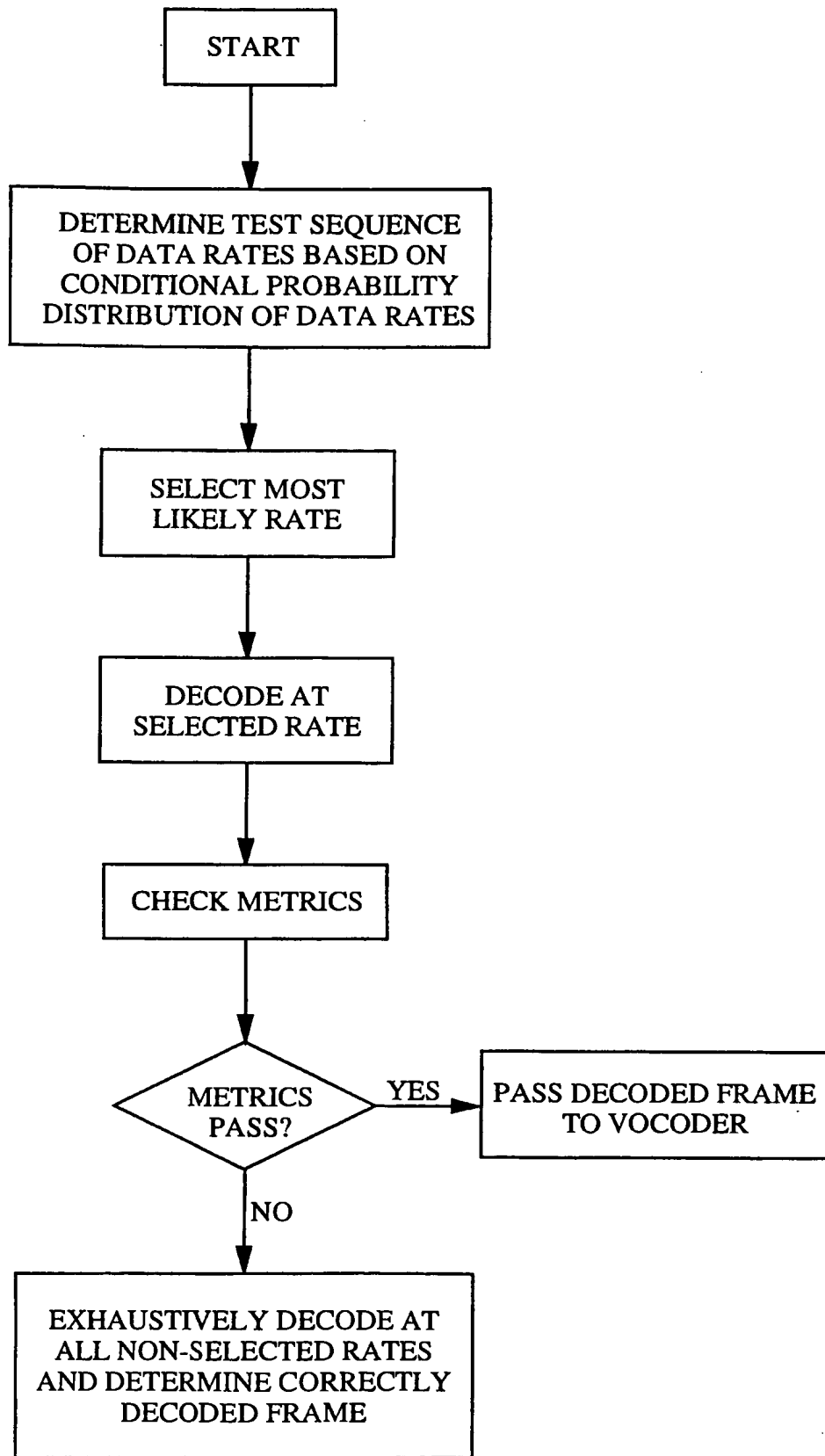


FIG. 8

INTERNATIONAL SEARCH REPORT

Inte. onal Application No

PCT/US 97/19676

A. CLASSIFICATION OF SUBJECT MATTER IPC 6 H04L25/02			
According to International Patent Classification (IPC) or to both national classification and IPC			
B. FIELDS SEARCHED			
Minimum documentation searched (classification system followed by classification symbols) IPC 6 H04L			
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched			
Electronic data base consulted during the international search (name of data base and, where practical, search terms used)			
C. DOCUMENTS CONSIDERED TO BE RELEVANT			
Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.	
A	EP 0 711 056 A (NIPPON ELECTRIC CO) 8 May 1996 see abstract see page 2, column 2, line 46 - page 3, column 3, line 26 see page 4, column 5, line 13 - column 6, line 23; figure 2 ---	1, 14, 15, 22, 27, 28	
A	EP 0 713 305 A (NIPPON ELECTRIC CO) 22 May 1996 see abstract see page 2, column 2, line 45 - page 3, column 3, line 12 see page 3, column 3, line 33 - column 4, line 2; figure 1 see page 3, column 4, line 44 - page 4, column 5, line 13 --- -/--	1, 14, 15, 22, 27, 28	
<input checked="" type="checkbox"/> Further documents are listed in the continuation of box C.		<input checked="" type="checkbox"/> Patent family members are listed in annex.	
* Special categories of cited documents :			
"A" document defining the general state of the art which is not considered to be of particular relevance		"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention	
"E" earlier document but published on or after the international filing date		"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone	
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)		"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.	
"O" document referring to an oral disclosure, use, exhibition or other means		"G" document member of the same patent family	
"P" document published prior to the international filing date but later than the priority date claimed			
Date of the actual completion of the international search <div style="text-align: center; font-weight: bold;">27 February 1998</div>		Date of mailing of the international search report <div style="text-align: center; font-weight: bold;">09/03/1998</div>	
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016		Authorized officer <div style="text-align: center; font-weight: bold;">Bossen, M</div>	

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 97/19676

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT		
Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 566 206 A (BUTLER BRIAN K ET AL) 15 October 1996 cited in the application see column 2, line 37 - line 47 see column 5, line 57 - column 6, line 32; figure 2 see column 7, line 35 - line 52; figure 4 -----	7,9-12, 14, 22-25,27

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

PCT/US 97/19676

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0711056 A	08-05-96	JP 8130535 A	21-05-96
		AU 3662595 A	09-05-96
		CA 2163134 A	03-05-96

EP 0713305 A	22-05-96	JP 2596392 B	02-04-97
		JP 8149567 A	07-06-96
		AU 686026 B	29-01-98
		AU 3787595 A	23-05-96
		CA 2162417 A	17-05-96
		FI 955398 A	17-05-96

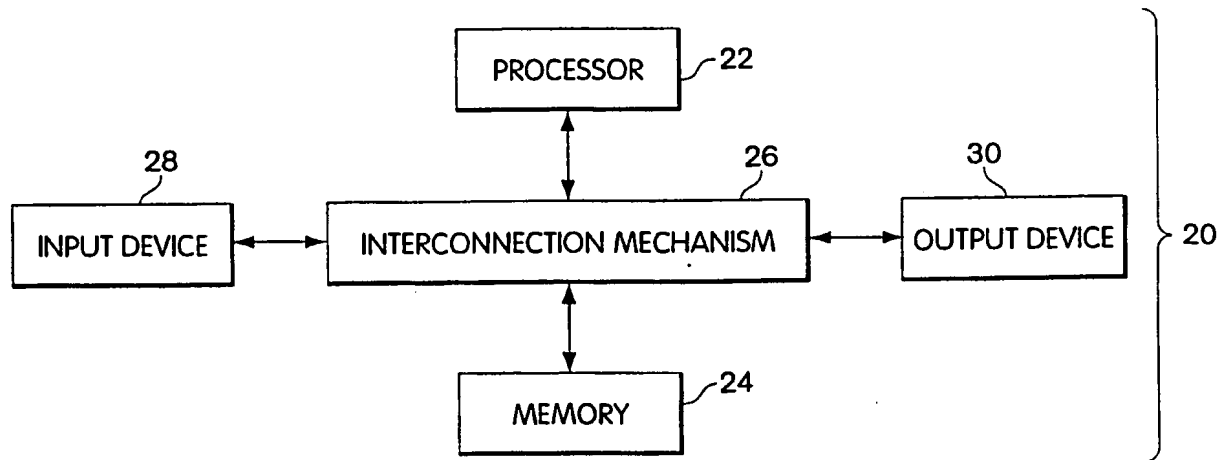
US 5566206 A	15-10-96	AT 158910 T	15-10-97
		AU 683479 B	13-11-97
		AU 7113694 A	17-01-95
		BR 9406891 A	26-03-96
		DE 69405997 D	06-11-97
		EP 0705512 A	10-04-96
		FI 956091 A	16-02-96
		JP 9501548 T	10-02-97
		WO 9501032 A	05-01-95
		CN 1108834 A	20-09-95
		IL 109842 A	30-09-97
		MX 9404610 A	31-01-95
		ZA 9404032 A	09-03-95



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<p>(51) International Patent Classification ⁶ : H04L 9/00</p>	<p>A1</p>	<p>(11) International Publication Number: WO 98/11690 (43) International Publication Date: 19 March 1998 (19.03.98)</p>
<p>(21) International Application Number: PCT/US97/16223 (22) International Filing Date: 12 September 1997 (12.09.97) (30) Priority Data: 60/025,991 12 September 1996 (12.09.96) US 08/887,723 3 July 1997 (03.07.97) US (71)(72) Applicant and Inventor: GLOVER, John, J. [US/US]; 26 Amaranth Avenue, Medford, MA 02155 (US). (74) Agent: GORDON, Peter, J.; Wolf, Greenfield & Sacks, P.C., 600 Atlantic Avenue, Boston, MA 02210 (US).</p>	<p>(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).</p> <p>Published <i>With international search report. Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</i></p>	

(54) Title: SELF-DECRYPTING DIGITAL INFORMATION SYSTEM AND METHOD



(57) Abstract

The claimed data protection device (20) includes a processor (22) connected to a memory system (24) through an interconnection mechanism (26). An input device (28) is also connected to the processor (22) and memory system (24) through the interconnection mechanism (26). The interconnection mechanism (26) is typically a combination of one or more buses and one or more switches. The output device (30) may be a display, and the input device (28) may be a keyboard and/or mouse or other cursor control device.

*(Referred to in PCT Gazette No. 32/1998, Section II)

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

SELF-DECRYPTING DIGITAL INFORMATION SYSTEM AND METHOD**Field of the Invention**

The present invention is related to mechanisms for protecting digital information from being copied. In particular, the present invention is related to mechanisms which permit authorized execution of computer program code or access to other digital information which is encrypted or otherwise encoded.

Background of the Invention

A serious problem which faces the electronic publishing and software industries is the ease with which digital information can be copied without authorization from the publisher. Digital information also may be used or modified without authorization. For example, computer software may be reverse engineered or attacked by computer viruses.

There are many mechanisms available which may be used to limit or prevent access to digital information. Such mechanisms often either restrict the ability of the user to make back-up copies or involve the use of special purpose hardware to limit access to the digital information. For example, some mechanisms restrict the use of digital information to a particular machine. See, for example, U.S. Patent 4,817,140. Other mechanisms require the digital information to be stored on a particular recording medium in order to be used. See, for example, U.S. Patent 5,412,718. Yet other mechanisms allow only a certain number of uses of the digital information. See for example, U.S. Patent 4,888,798. Many of these access control mechanisms cause distribution to be more costly.

Several other patents describe a variety of systems for encryption, compression, licensing and royalty control and software distribution such as: U.S. Pat. No. 4,405,829, U.S. Pat. No. 4,864,616, U.S. Pat. No. 4,888,800, U.S. Pat. No. 4,999,806, U.S. Pat. No. 5,021,997, U.S. Patent No. 5,027,396, U.S. Pat. No. 5,033,084, U.S. Pat. No. 5,081,675, U.S. Pat. No. 5,155,847, U.S. Pat. No. 5,166,886, U.S. Pat. No. 5,191,611, U.S. Pat. No. 5,220,606, U.S. Pat. No. 5,222,133, U.S. Pat. No. 5,272,755, U.S. Pat. No. 5,287,407, U.S. Pat. No. 5,313,521, U.S. Pat. No. 5,325,433, U.S. Pat. No. 5,327,563, U.S. Pat. No. 5,337,357, U.S. Pat. No. 5,351,293, U.S. Pat. No. 5,341,429, U.S. Pat. No. 5,351,297, U.S. Pat. No. 5,361,359, U.S. Pat. No. 5,379,433, U.S. Pat. No. 5,392,351, U.S. Pat. No. 5,394,469, U.S. Pat. No. 5,414,850, U.S. Pat. No. 5,473,687, U.S. Pat. No. 5,490,216, U.S. Pat. No. 5,497,423, U.S. Pat. No. 5,509,074, U.S.

Pat. No. 5,511,123, U.S. Pat. No. 5,524,072, U.S. Pat. No. 5,532,920, U.S. Pat. No. 5,555,304, U.S. Pat. No. 5,557,346, U.S. Pat. No. 5,557,765, U.S. Pat. No. 5,592,549, U.S. Pat. No. 5,615,264, U.S. Pat. No. 5,625,692, and U.S. Pat. No. 5,638,445.

Computer programs or other digital information also may be encrypted in order to prevent an individual from making a useful copy of the information or from reverse engineering a program. Even with such encryption, however, a computer program must be decrypted in order for a computer to load and execute the program. Similarly, other digital information must be decrypted before it can be accessed and used. Generally, digital information is decrypted to disk, and not to main memory of the computer which is more protected by the operating system, because decryption to main memory results in a significant loss of memory resources. If the purpose for using encryption is to prevent users from copying the digital information, then decryption of the information to accessible memory for use defeats this purpose.

One way to protect digital information using encryption has been made available by International Business Machines (IBM) and is called a "CRYPTOLOPE" information container. This technology is believed to be related to U.S. Patent Nos. 5,563,946 and 5,598,470 (to Cooper et al.), and published European patent applications 0679977, 0679978, 0679979 and 0681233. The CRYPTOLOPE system requires a user to have a "helper application" and a key. The CRYPTOLOPE information container is generated by IBM. The content provider submits data to IBM, which in turn encrypts and packages the data in a CRYPTOLOPE information container. The helper application is a form of memory resident program, called a terminate and stay resident (TSR) program, which is a form of input/output (I/O) device driver installed in the operating system and which monitors requests from the operating system for files on specified drives and directories. Because the TSR program must know the directory, and/or file name to be accessed, that information also is available to other programs. Other programs could use that information to manipulate the operation of the TSR program in order to have access to decrypted contents of the information container. The encrypted information container includes an executable stub which is executed whenever the application is run without the installed TSR program or from a drive not monitored by the TSR program to prevent unpredictable activity from executing encrypted code. This stub may be used to install decryption and cause the application be executed a second time, or to communicate with the TSR program to instruct the TSR program to monitor the drive. It may be preferable from the point of view of the content provider however to maintain an encryption process and keys independently of any third party.

Multimedia content, such as a movie or hypertext presentation also may be stored on a digital versatile disk (DVD), sometimes called a digital video disk, compact disk read-only memory (CD-ROM), rewriteable compact disks (CD-RW) or other medium in an encrypted digital format for use with special-purpose devices. For example, concern about illegal copying
5 of content from digital video disks or other digital media has resulted in a limited amount of content being available for such devices. This problem has caused representatives of both multimedia providers and digital video disk manufacturers to negotiate an agreement on an encryption format for information stored on DVDs. This copy protection scheme is licensed through an organization called the CSS Interim Licensing organization. However, in this
10 arrangement, the content provider is limited to using the agreed upon encryption format and a device manufacturer is limited to using a predetermined decryption system.

Encryption has also been used to protect and hide computer viruses. Such viruses are typically polymorphic, i.e., they change every time they infect a new program, and are encrypted. The virus includes a decryption program that executes to decrypt the virus every time the
15 infected program is run. Such viruses are described, for example, in "Computer Virus-Antivirus Coevolution" by Carey Nachenberg, Communications of the ACM, Vol. 40, No. 1, (Jan. 1997), p. 46 et seq. Such viruses include decryption keys within them since, clearly, their execution is not carried out by the user and a user would not be asked for authorization keys to permit execution of the viruses. Additionally, such viruses are typically only executed once at the start
20 of execution of an infected program and permanently return control to the infected program after execution.

Summary of the Invention

Some of these problems with digital information protection systems may be overcome
25 by providing a mechanism which allows a content provider to encrypt digital information without requiring either a hardware or platform manufacturer or a content consumer to provide support for the specific form of corresponding decryption. This mechanism can be provided in a manner which allows the digital information to be copied easily for back-up purposes and to be transferred easily for distribution, but which should not permit copying of the digital information
30 in decrypted form. In particular, the encrypted digital information is stored as an executable computer program which includes a decryption program that decrypts the encrypted information

to provide the desired digital information, upon successful completion of an authorization procedure by the user.

In one embodiment, the decryption program is executed as a process within a given operating system and decrypts the digital information within the memory area assigned to that process. This memory area is protected by the operating system from copying or access by other processes. Even if access to the memory area could be obtained, for example through the operating system, when the digital information is a very large application program or a large data file, a copy of the entire decrypted digital information is not likely to exist in the memory area in complete form.

By encrypting information in this manner, a platform provider merely provides a computer system with an operating system that has adequate security to define a protected memory area for a process and adequate functionality to execute a decryption program. The content provider in turn may use any desired encryption program. In addition, by having a process decrypt information within a protected memory area provided by the operating system, the decrypted information does not pass through any device driver, memory resident program or other known logical entity in the computer system whose behavior may be controlled to provide unauthorized access to the data. The ability to reverse engineer or attack a computer program with a computer virus also may be reduced.

In another embodiment, the decryption program is part of a dynamically loaded device driver that responds to requests for data from the file containing the encrypted data. When the digital information product is first executed, this device driver is extracted from the file and is loaded into the operating system. The executed digital information product then informs the loaded device driver of the location of the hidden information in the file, any keys or other passwords, and the name of a phantom directory and file to be called that only the digital information product and the device driver know about. The name of this directory may be generated randomly. Each segment of hidden information in the digital information product may be assigned its own unique file name in the phantom directory. The digital information product then makes a call to the operating system to execute one of the files in the phantom directory. The loaded driver traps these calls to the operating system, accesses the original file, decrypts the desired information and outputs the desired information to the operating system.

In combination with other mechanisms that track distribution, enforce royalty payments and control access to decryption keys, the present invention provides an improved method for

identifying and detecting sources of unauthorized copies. Suitable authorization procedures also enable the digital information to be distributed for a limited number of uses and/or users, thus enabling per-use fees to be charged for the digital information.

Accordingly, one aspect of the invention is a digital information product including a
5 computer-readable medium with digital information stored thereon. The digital information includes computer program logic having a first portion of executable computer program logic and a second portion of digital information. The first portion of executable program logic, when executed, defines a mechanism for responding to requests for digital information from an operating system of a computer. This mechanism, when used to access the second portion of the
10 encrypted digital information, decrypts the encrypted digital information, and provides the encrypted digital information to the operating system.

In the foregoing aspect of the invention, the digital information may be executable computer program logic. Hence, one aspect of the invention is a computer program product, including a computer readable medium with computer program logic stored thereon. The
15 computer program logic includes a first portion of executable computer program logic and a second portion of encrypted computer program logic. The first portion of executable computer program logic, when executed, defines a mechanism for responding to requests for computer program logic from an operating system of a computer. This mechanism accesses the second portion of encrypted computer program logic, decrypts the encrypted computer program logic,
20 and provides the decrypted computer program logic to the operating system.

Another aspect of the present invention is a computer program product, a computer system and a process which produce a computer program or digital information product in accordance with other aspects of the invention, using executable program code for the first and second portions of the desired computer program product.

25 Another aspect of the present invention is a computer program product including a self-decrypting encrypted executable computer program. The product includes a computer readable medium having computer program logic stored thereon. The computer program logic defines first, second and third modules, wherein the third module defines the encrypted executable computer program. The first module, when executed by a computer, defines a mechanism for
30 loading the second module into memory of the computer. The second module, when executed by a computer, defines a mechanism for communicating with an operating system of the computer to receive requests for program code from the encrypted executable computer program from the

third module, and for processing the requests to access and decrypt the encrypted executable computer program and for providing the decrypted executable code from the third module to the operating system.

Another aspect of the invention is a process for executing encrypted executable
5 computer programs on a computer system having a processor, memory and operating system. The process involves receiving computer program logic having a first module defining a start up routine, a second module, and a third module containing the encrypted executable computer program. The first module of the received computer program logic is executed using the processor. When the first module is executed, the second module is caused to be loaded into the
10 memory of the computer system. Requests are generated from the operating system for data from the encrypted executable computer program and are received by the second module. The second module accesses and decrypts the encrypted executable computer program in response to these requests and returns the decrypted executable computer program to the operating system.

These and other aspects, advantages and features of the present invention and its
15 embodiments will be more apparent given the following detailed description.

Brief Description of the Drawing

In the drawing,

Fig. 1 is a block diagram of a typical computer system with which the present invention
20 may be implemented;

Fig. 2 is a block diagram of a memory system in the computer system of Fig. 1;

Fig. 3 is a diagram of a computer program or digital information product which may be recorded on a computer readable and writable medium, such as a magnetic disc;

Fig. 4 is a flowchart describing how the computer program or digital information
25 product of Fig. 3 is used;

Fig. 5 is a flowchart describing operation of an example unwrap procedure as shown in Fig. 3 in one embodiment of the invention;

Fig. 6 is a flowchart describing operation of an example device driver as shown in Fig. 3 in one embodiment of the invention;

Fig. 7 is a block diagram of a computer system in the process of executing a computer
30 program product in accordance with one embodiment of the invention;

Fig. 8 is a flowchart describing operation of an example unwrap procedure in another embodiment of the invention; and

Fig. 9 is a flowchart describing how a computer program product such as shown in Fig. 3 is constructed.

5

Detailed Description

The present invention will be more completely understood through the following detailed description which should be read in conjunction with the attached drawing in which similar reference numbers indicate similar structures.

10 Embodiments of the present invention may be implemented using a general purpose digital computer or may be implemented for use with a digital computer or digital processing circuit. A typical computer system 20 is shown in Fig. 1, and includes a processor 22 connected to a memory system 24 via an interconnection mechanism 26. An input device 28 also is connected to the processor and memory system via the interconnection mechanism, as is an
15 output device 30. The interconnection mechanism 26 is typically a combination of one or more buses and one or more switches. The output device 30 may be a display and the input device may be a keyboard and/or a mouse or other cursor control device.

It should be understood that one or more output devices 30 may be connected to the computer system. Example output devices include a cathode ray tube (CRT) display, liquid
20 crystal display (LCD), television signal encoder for connection to a television or video tape recorder, printers, communication devices, such as a modem, and audio output. It also should be understood that one or more input devices 28 may be connected to the computer system. Example input devices include a keyboard, keypad, trackball, mouse, pen and tablet, communication device, audio or video input and scanner. It should be understood that the
25 invention is not limited to the particular input or output devices used in combination with the computer system or to those described herein.

The computer system 20 may be a general purpose computer system, which is programmable using a high level computer programming language, such as "C++," "Pascal," "VisualBasic." The computer system also may be implemented using specially programmed,
30 special purpose hardware. In a general purpose computer system, the processor is typically a commercially available processor, such as the Pentium processor from Intel Corporation. Many other processors are also available. Such a processor executes a program called an operating

system, such as Windows 95 or Windows NT 4.0, both available from Microsoft Corporation, which controls the execution of other computer programs and provides scheduling, debugging, input output control, accounting compilation, storage assignment, data management and memory management, and communication control and related services. Other examples of operating systems include: MacOS System 7 from Apple Computer, OS/2 from IBM, VMS from Digital Equipment Corporation, MS-DOS from Microsoft Corporation, UNIX from AT&T, and IRIX from Silicon Graphics, Inc.

The computer system 20 also may be a special purpose computer system such as a digital versatile disk or digital video disk (DVD) player. In a DVD player, there is typically a decoder controlled by some general processor which decodes an incoming stream of data from a DVD. In some instances, the DVD player includes a highly integrated DVD decoder engine. Such devices generally have a simple operating system which may be modified to include the capabilities described and used herein in connection with the typical operating systems in a general purpose computer. In particular, some operating systems are designed to be small enough for installation in an embedded system such as a DVD player, including the WindowsCE operating system from Microsoft Corporation and the JavaOS operating system from SunSoft Corporation. The operating system allows a content provider to provide its own programs that define some of the content, which is particularly useful for interactive multimedia. This capability also can be used to provide encryption and decryption, in accordance with the invention.

The processor and operating system define a computer platform for which application programs in a programming language such as an assembly language or a high level programming language are written. It should be understood that the invention is not limited to a particular computer platform, operating system, processor, or programming language. Additionally, the computer system 20 may be a multi-processor computer system or may include multiple computers connected over a computer network.

An example memory system 24 will now be described in more detail in connection with Fig. 2. A memory system typically includes a computer readable and writable non-volatile recording medium 40, of which a magnetic disk, a flash memory, rewriteable compact disk (CD-RW) and tape are examples. The recording medium 40 also may be a read only medium such as a compact disc-read only memory (CD-ROM) or DVD. A magnetic disk may be removable, such as a "floppy disk" or "optical disk," and/or permanent, such as a "hard drive." The disk,

which is shown in Fig. 2, has a number of tracks, as indicated at 42, in which signals are stored, in binary form, i.e., a form interpreted as a sequence of 1's and 0's, as shown at 44. Such signals may define an application program to be executed by the microprocessor, or information stored on the disk to be processed by the application program. Typically, in the operation of a general purpose computer, the processor 22 causes data to be read from the non-volatile recording medium 40 into an integrated circuit memory element 46, which is typically a volatile random access memory, such as a dynamic random access memory (DRAM) or static random access memory (SRAM). The integrated circuit memory element 46 allows for faster access to the information by the processor than disk 40, and is typically called the system or host memory. The processor generally causes the data to be manipulated within the integrated circuit memory 46 and may copy the data to the disk 40, if modified, when processing is completed. A variety of mechanisms are known for managing data movement between the disk 40 and the integrated circuit memory 46, and the invention is not limited thereto. It should also be understood that the invention is not limited to a particular memory system.

The file system of a computer generally is the mechanism by which an operating system manages manipulation of data between primary and secondary storage, using files. A file is a named logical construct which is defined and implemented by the operating system to map the name and a sequence of logical records of data to physical storage media. An operating system may specifically support various record types or may leave them undefined to be interpreted or controlled by application programs. A file is referred to by its name by application programs and is accessed through the operating system using commands defined by the operating system. An operating system provides basic file operations provided by for creating a file, opening a file, writing a file, reading a file and closing a file.

In order to create a file, the operating system first identifies space in the storage media which is controlled by the file system. An entry for the new file is then made in a directory which includes entries indicating the names of the available files and their locations in the file system. Creation of a file may include allocating certain available space to the file. Opening a file returns a handle to the application program which it uses to access the file. Closing a file invalidates the handle.

In order to write data to a file, an application program issues a command to the operating system which specifies both an indicator of the file, such as a file name, handle or other descriptor, and the information to be written to the file. Given the indicator of the file, the

operating system searches the directory to find the location of the file. The directory entry stores a pointer, called the write pointer, to the current end of the file. Using this pointer, the physical location of the next available block of storage is computed and the information is written to that block. The write pointer is updated in the directory to indicate the new end of the file.

5 In order to read data from a file, an application program issues a command to the operating system specifying the indicator of the file and the memory locations assigned to the application where the next block of data should be placed. The operating system searches its directory for the associated entry given the indicator of the file. The directory may provide a pointer to a next block of data to be read, or the application may program or specify some offset
10 from the beginning of the file to be used.

A primary advantage of using a file system is that, for an application program, the file is a logical construct which can be created, opened, written to, read from and closed without any concern for the physical storage used by the operating system.

The operating system also allows for the definition of another logical construct called a
15 process. A process is a program in execution. Each process, depending on the operating system, generally has a process identifier and is represented in an operating system by a data structure which includes information associated with the process, such as the state of the process, a program counter indicating the address of the next instruction to be executed for the process, other registers used by process and memory management information including base and bounds
20 registers. Other information also may be provided. The base and bounds registers specified for a process contain values representing the largest and smallest addresses that can be generated and accessed by an individual program. Where an operating system is the sole entity able to modify these memory management registers, adequate protection from access to the memory locations of one process from another process is provided. As a result, this memory management information
25 is used by the operating system to provide a protected memory area for the process. A process generally uses the file system of the operating system to access files.

The present invention involves storing encrypted digital information, such an audio, video, text or an executable computer program, on a computer readable medium such that it can be copied easily for back-up purposes and transferred easily for distribution, but also such that it
30 cannot be copied readily in decrypted form during use. In particular, the digital information is stored as a computer program that decrypts itself while it is used to provide the digital information, e.g., to provide executable operation code to the operating system of a computer, as

the digital information is needed. Any kind of encryption or decryption may be used and also may include authorization mechanisms and data compression and decompression. In one embodiment of the present invention, decrypted digital information exists only in memory accessible to the operating system and processes authorized by the operating system. When the digital information is a large application program, a copy of the entire decrypted application program is not likely to exist in the main memory at any given time, further reducing the likelihood that a useful copy of decrypted code could be made. The decryption operation also is performed only if some predetermined authorization procedure is completed successfully.

One embodiment of the invention, in which the decryption program is a form of dynamically loaded device driver, will first be described. Fig. 3 illustrates the structure of digital information as stored in accordance with one embodiment of the present invention, which may be stored on a computer readable medium such as a magnetic disc or compact disc read only memory (CD-ROM) to form a computer program product. The digital information includes a first portion 50, herein called an unwrap procedure or application, which is generally unencrypted executable program code. The purpose of the unwrap procedure is to identify the locations of the other portions of the digital information, and may perform other operations such as verification. In particular, the unwrap procedure identifies and extracts a program which will communicate with the operating system, herein called a virtual device driver 52. The unwrap procedure may include decryption and decompression procedures to enable it to decrypt/decompress the driver, and/or other content of this file. The program 52 need not be a device driver. The virtual device driver 52 typically follows the unwrap procedure 50 in the file container, the digital information. The virtual device driver, when executed, decrypts and decodes the desired digital information such as an executable computer program code from hidden information 54, which may be either encrypted and/or encoded (compressed). It is the decrypted hidden information which is the desired digital information to be accessed. This hidden information may be any kind of digital data, such as audio, video, text, and computer program code including linked libraries or other device drivers.

In this embodiment of the computer program product, labels delineate the boundaries between the device driver and the hidden files. These labels may or may not be encrypted. A first label 56 indicates the beginning of the code for the virtual device driver 52. A second label 58 indicates the end of the virtual device driver code. Another label 60 indicates the beginning of the hidden information and a label 62 indicates the end of that application. There may be one

or more blocks of such hidden information, each of which can be given a different name. It may be advantageous to use the name of the block of information in its begin and end tags. This computer program product thus contains and is both executable computer program code and one or more blocks of digital information. A table of locations specifying the location of each
5 portion of the product could be used instead of labels. Such a table could be stored in a predetermined location and also may be encrypted.

The overall process performed using this computer program product in one embodiment of the invention will now be described in connection with Fig. 4. This embodiment may be implemented for use with the Windows95 operating system and is described in more detail in
10 connection with Figs. 5-7. An embodiment which may be implemented for use on the WindowsNT 4.0 operating system is described in more detail below in connection with Fig. 8. In both of these described embodiments, the digital information is an executable computer program which is read by the operating system as data from this file and is executed. The same principle of operation would apply if the data were merely audio, video, text or other information
15 to be conveyed by a user. In the embodiment of Fig. 4, the computer program is first loaded into memory in step 70, and the unwrap procedure 50 is executed by the operating system, as any typical executable computer program is executed. The unwrap procedure may perform authorization, for example by checking for a required password or authentication code, and may receive any data needed for decryption or decompression, for example keys or passwords, in step
20 72. Suitable authorization procedures may provide the ability to distribute software for single use. The unwrap procedure locates the virtual device driver 52 within the computer program in step 74, and then locates the hidden application in step 76. The virtual device driver 52 is then extracted by the unwrap procedure from the computer program, copied to another memory location and loaded for use by the operating system in step 78. An advantage of an operating
25 system like Windows95 is that it allows such device drivers to be loaded dynamically without restarting the computer.

The executed unwrap procedure 50, in step 80, informs the loaded virtual device driver 52 of the location of the hidden information in the file, any keys or other passwords, and a name of a phantom directory and file to be called that only the unwrap procedure and the virtual device
30 driver know about. The name of this phantom directory may be generated randomly. Each segment information hidden in the digital information product may be assigned its own unique file name in the phantom directory.

After the loaded virtual device driver 52 receives all communications from the unwrap procedure, it opens the original application file for read only access in step 82. The unwrap procedure then makes a call to the operating system in step 84 to execute the file in the phantom directory for which the name was transmitted to the loaded virtual device driver. One function of the loaded virtual device driver 52 is to trap all calls from the operating system to access files in step 86. Any calls made by the operating system to access files in the phantom directory are processed by the virtual device driver, whereas calls to access files in other directories are allowed to proceed to their original destination. In response to each call from the operating system, the virtual device driver obtains the bytes of data requested by the operating system from the original computer program file in step 88. These bytes of data are then decrypted or decompressed in step 90 and returned to the operating system. When processing is complete, the phantom application is unloaded from the operating system in step 92, and may be deleted from the memory.

A more detailed description of the process of Fig. 4 will now be described in connection with Figs. 5-7. Fig. 5 is a flowchart describing the operation of one embodiment of the unwrap procedure in more detail. The first step performed by this procedure is identifying the operating system being used, in step 100. This step is useful because different methods may be used with different operating systems. All code that may be used to run in various operating systems may be placed in this unwrap procedure. This procedure also may contain the decompression/decryption code, for example or any other computer program code to be executed.

The executed application then opens the original executable file as a data file and searches for the begin and end tags of the device driver and hidden files in step 102. The device driver code is copied into memory and loaded into the operating system in step 104. The unwrap procedure then informs the device driver of the name of the original application file, offsets of the hidden files and the name of a phantom directory, which is typically randomly generated (step 106). This communication may be performed using a "DeviceIOControl" function call in the Windows95 operating system. The unwrap procedure then makes a call to the operating system to execute the hidden file in the phantom directory, in step 108.

The operation of one embodiment of a device driver will now be described in connection with Fig. 6. After the device driver is loaded into the operating system, it hooks into a position between the operating system and a file system driver (FSD), in step 110, to intercept

calls made by the operating system to the FSD for data from files in the phantom directory. The FSD is the code within the operating system that performs physical reading and writing of data to disk drives. The operating system makes requests to the FSD for data from files in directories on the disk drives. The driver then receives information from the unwrap procedure including the
5 name of the original file, the location of hidden files within the original file, and the name of the phantom directory created by the unwrap procedure (step 112). The device driver opens the original file as a read only data file. The device driver now traps calls, in step 114, made from the operating system for files in the phantom directory. Calls to other directories are ignored and passed on to the original destination. The device driver then reads the data from the original data
10 file, decrypts and decompresses it, and returns the decrypted/decompressed data to the operating system in step 116.

For example, if the offset for the hidden application in the original data file is 266,270 bytes and the operating system asks for 64 bytes starting at offset 0 of the hidden application in the phantom directory, the device driver reads 64 bytes from the original file starting at offset
15 266,270, decrypts/decompresses those 64 bytes, and returns the first 64 decrypted/decompressed bytes back to the operating system. From the point of view of the operating system, the 64 bytes appear to have come from the file in the phantom directory. Steps 114 and 116 are performed on demand in response to the operating system.

A block diagram of the computer system in this embodiment, with a device driver
20 loaded and in operation, will now be described in more detail in connection with Fig. 7. Fig. 7 illustrates the operating system 120, the loaded device driver 122, a file system driver 124, the original executable file 126 as it may appear on disk and the unwrap procedure 128. The executable file may in fact be on a remote computer and accessed through a network by the device driver. The unwrap procedure causes the operating system to begin execution of the
25 hidden file by issuing an instruction to execute the file in the phantom directory, as indicated at 130. This command is issued after the device driver 122 is informed of the file name of the original executable file 126, offsets of the hidden files within that file and the name of the phantom directory, as indicated at 132. The operating system then starts making calls to the phantom directory as indicated at 134. The device driver 122 traps these calls and turns them
30 into requests 136 to the file system driver to access the original executable file 126. Such requests actually are made to the operating system 120, through the device driver 122 to the file system driver 124. The file system driver 124 returns encrypted code 138 to the device driver

122. The encrypted code 138 actually passes back through the device driver 122 to the operating system 120 which in turn provides the encrypted code 138 to the device driver 122 as the reply to the request 136 for the original file. The device driver 122 then decrypts the code to provide decrypted code 140 to the operating system 120.

5 Another embodiment of the invention will now be described in connection with Fig. 8. This embodiment may be implemented using the WindowsNT 4.0 operating system, for example. In this embodiment, the device driver portion 52 of the computer program product is not used. The unwrap procedure for this embodiment begins by identifying the operating system being used similar, which is step 100 in Fig. 5. If the operating system is Windows NT 4.0, for
10 example, a different unwrap procedure for this embodiment is performed. Before describing this unwrap procedure, a brief description of some of the available operating system commands will be provided.

 Currently, under all versions of the Window operating system or operating environment from Microsoft Corporation (such as Windows 3.1, Windows 95 and Windows NT 3.51 and 4.0)
15 all executable files (.exe) or dynamic link library (.dll and .ocx) files, which are executable files with different header and loading requirements than .exe files, that are loaded into memory by the operating system must reside as a file either locally, e.g., on a disk drive or remotely, e.g., over a network or communications port. All further references herein to loading an executable will be using the Win32 function calls used in Windows 95 and NT 3.51 and 4.0 operating
20 systems. The CreateProcess() function which loads files with an .exe extension takes ten parameters:

```

BOOL CreateProcess(// Prototype from Microsoft Visual C++ Help Documentation
    LPCTSTR lpApplicationName,           // pointer to name of executable module
    25  LPTSTR lpCommandLine,           // pointer to command line string
    LPSECURITY_ATTRIBUTES lpProcessAttributes, // pointer to process security attributes
    LPSECURITY_ATTRIBUTES lpThreadAttributes, // pointer to thread security attributes
    BOOL bInheritHandles,              // handle inheritance flag
    DWORD dwCreationFlags,            // creation flags
    30  LPVOID lpEnvironment,          // pointer to new environment block
    LPCTSTR lpCurrentDirectory,       // pointer to current directory name
    LPSTARTUPINFO lpStartupInfo,      // pointer to STARTUPINFO
    LPPROCESS_INFORMATION lpProcessInformation // pointer to PROCESS_INFORMATION
);

```

Three of these parameters are pointers to strings that contain an application file name, command line parameters, and the current directory. The other parameters are security, environmental, and process information. The LoadLibrary() function takes one parameter that is a pointer to a string that contains the application file name:

5

```
HINSTANCE LoadLibrary(// Prototype from Microsoft Visual C++ Help Documentation
    LPCTSTR lpLibFileName    // address of filename of executable module
);
```

10 The LoadLibraryEx() function takes three parameters the first being the same as LoadLibrary(), the second parameter must be null, and the third tells the operating system whether to load the file as an executable or as a data file in order to retrieve resources such as icons or string table data from it and not load it as an executable:

```
15 HINSTANCE LoadLibraryEx(// Prototype from Microsoft Visual C++ Help Documentation
    LPCTSTR lpLibFileName,    // points to name of executable module
    HANDLE hFile,            // reserved, must be NULL
    DWORD dwFlags           // entry-point execution flag
);
```

20

The CreateFile() function is used to create and open files and to load files such as device drivers. This function also requires a pointer to a string that contains the name of a physical file:

```
HANDLE CreateFile(// Prototype from Microsoft Visual C++ Help Documentation
25 LPCTSTR lpFileName,                // pointer to name of the file
    DWORD dwDesiredAccess,          // access (read-write) mode
    DWORD dwShareMode,              // share mode
    LPSECURITY_ATTRIBUTES lpSecurityAttributes, // pointer to security descriptor
    DWORD dwCreationDistribution,    // how to create
30 DWORD dwFlagsAndAttributes,      // file attributes
    HANDLE hTemplateFile           // handle to file with attributes to copy
);
```

There are other functions such as `MapViewOfFile()` and `MapViewOfFileEx()` that map areas of memory to an already opened physical file through a handle to that file. They have the following parameters:

```

5  LPVOID MapViewOfFile(// Prototype from Microsoft Visual C++ Help Documentation
    HANDLE hFileMappingObject,           // file-mapping object to map into address space
    DWORD dwDesiredAccess,              // access mode
    DWORD dwFileOffsetHigh,             // high-order 32 bits of file offset
    DWORD dwFileOffsetLow,              // low-order 32 bits of file offset
10  DWORD dwNumberOfBytesToMap          // number of bytes to map
    );

```

```

    LPVOID MapViewOfFileEx(// Prototype from Microsoft Visual C++ Help Documentation
    HANDLE hFileMappingObject,           // file-mapping object to map into address space
15  DWORD dwDesiredAccess,              // access mode
    DWORD dwFileOffsetHigh,             // high-order 32 bits of file offset
    DWORD dwFileOffsetLow,             // low-order 32 bits of file offset
    DWORD dwNumberOfBytesToMap,        // number of bytes to map
    LPVOID lpBaseAddress                // suggested starting address for mapped view
20  );

```

All of the foregoing functions directly use a pointer to a string that is a physical file. The only file functions that do not directly use a physical filename are functions like `CreateNamedPipe()`, which has the following parameters:

```

25  HANDLE CreateNamedPipe(// Prototype from Microsoft Visual C++ Help Documentation
    LPCTSTR lpName,                      // pointer to pipe name
    DWORD dwOpenMode,                    // pipe open mode
    DWORD dwPipeMode,                    // pipe-specific modes
    DWORD nMaxInstances,                  // maximum number of instances
30  DWORD nOutBufferSize,                 // output buffer size, in bytes
    DWORD nInBufferSize,                 // input buffer size, in bytes
    DWORD nDefaultTimeOut,               // time-out time, in milliseconds
    LPSECURITY_ATTRIBUTES lpSecurityAttributes // pointer to security attributes structure
    );
35

```

The string to which CreateNamedPipe() points using the first parameter is a string that both an existing executable and the operating system know about and does not exist physically.

Unfortunately both of the executables that "know" this private name could only be loaded using one of the other procedures that required a physical file. Currently it is not possible to load an
5 executable using a "named pipe" name. Both of or any executables that use the name of the "named pipe" already must have been loaded into memory.

All of the foregoing functions require a physical file because all of them use "file mapping" processes. File mapping allows large executable files to appear to be loaded rapidly since they are rarely completely loaded into memory but rather are mapped into memory. The
10 detriment to this mapping capability is that executable code must remain in physical memory in a file in unencrypted form in order to be loaded, unless there is a middle layer or file system driver that the operating system uses as a physical layer and that decrypts the executable code to the operating system on demand. The potential weakness here is that another file system driver can
hook into the operating system to monitor traffic between the operating system and all file
15 system drivers and capture decrypted executable code passing from the file system driver to the operating system. Some operating systems allow such monitoring more than others. Many anti-viral software packages use this technique to prevent computer virus attacks.

One method of loading and executing encrypted executable computer program code is to use a stub executable having two parts. The first part is the normal front end loader code that all
20 executables have. In addition, the first part would perform any authorization which may include receiving a password from the user, then allocate enough memory to hold hidden encrypted code when it is decrypted, either in its entirety or a portion of it, copy the encrypted code into that area of protected (and preferably locked so no disk swapping occurs) memory, decrypt it once it is in
memory and only in memory, and then have the operating system load the code only from
25 memory therefore bypassing any file system drivers or TSRs so they have access to only encrypted code.

Some of the file functions listed above and similar functions on other operating systems could be modified easily by a programmer having access to source code for those operating
systems, or a new operating system may be made to provide functions which allow direct loading
30 of executable code from memory rather than physical files. For example, in the Win32 commands, a command similar to CreateProcess() command could be provided. The command should have a few extra parameters including the process identifier of the process that contains

the now decrypted executable code, the memory address of the start of the decrypted code, and the size of the decrypted code. The command could also contain a parameter specifying a "call back" function within the first process that would provide decrypted code on demand directly to the operating system through a protected buffer, therefore allowing only a portion of the encrypted code to be decrypted at any one time instead of in its entirety, for better protection and less memory use. The second parameter of the LoadLibraryEx() command that now needs to be NULL could be expanded to hold a structure that contained the same information. Both of these and other similar functions could be changed or created to allow loading executable code either as an .exe, .dll, or other extensions or identifiers, such as by using a "named pipe" name that only the operating system and process that holds decrypted code know about and having the operating system load from the named pipe.

Alternatively, without having such additional capabilities in the operating system, an application program can be divided into two parts. The first part is code that is common to all applications such as code for allocating memory off the heap and code that provides some interaction with the user. This kind of code is generally not code that the content provider is concerned about copying. The second part is the code that the content provider believes is valuable. Typically this valuable code is a business logic code or what would be considered a middle tier of a three-tier environment. A content provider would like to protect this second part of the code, at least much more than the first part of the code. The content provider would place all of the important code to be protected inside a dynamic link library and the code that is not that important would reside in the front end "stub" executable. Both of these would be combined into another executable containing the .dll in encrypted form only, along with any other files, data, information, and/or tables for holding, for example, hardware identifiers. This other executable is the final digital information product.

The first part of the digital information product, i.e., the executable stub, would load and execute normally like any other application. It then would perform any authorization procedures. Once the proper authorization or password was completed successfully, an unwrap procedure would be performed as will now be described in connection with Fig. 8, it would then allocate enough protected memory using a function like VirtualAlloc() as shown in step 150:

30

```
DWORD nFileSize = 0;  
DWORD nPhantomFileSize = 0;
```

- 20 -

```

DWORD exeOffset = 0;
DWORD nPreferredLoadAddress = GetPreCompiledLoadAddress();
CString cCommandFile = UnwrapGetNTCommandFile();
exeOffset = UnwrapGetDllOffset(cCommandFile);
5  nFileSize = UnwrapGetDllSize(cCommandFile);
   nPhantomFileSize = nFileSize + 0x3000; // add any needed extra space
   // Increase buffer size to account for page size (currently Intel page size).
   DWORD nPageSize = GetPageSize();
   nPhantomFileSize += (nPageSize - (nPhantomFileSize % nPageSize));
10 // Allocate the memory to hold the decrypted executable.
   LPVOID lpvBlock = VirtualAlloc((LPVOID) nPreferredLoadAddress,
                                nPhantomFileSize,
                                MEM_RESERVE | MEM_COMMIT, PAGE_READWRITE);

15 This function can request a particular address space. Preferably, this address space is the
   preferred load address space to which the .dll was linked in order to minimize any needed
   relocation and fix up code. The stub executable may lock that area of memory in step 152, for
   example by using VirtualLock() to prevent any memory writes to a swap file, depending on the
   operating system, as shown below:

20  BOOL bVLock = VirtualLock((LPVOID) nPreferredLoadAddress, nPhantomFileSize);

   The memory area still should be secure even without this preventive step since the Windows 95
   and NT operating systems do not allow any user access to swap files.

25  The encrypted code is then copied from the digital information product into the allocated
   protected memory in step 154, for example by using the following command:

   UnwrapCopyHiddenExeToMem(cCommandFile, exeOffset, nFileSize, (char *) lpvBlock);

30  Once in memory, the stub would then decrypt the code to that same portion of memory in step
   156, for example by using the following commands:

```



```
CwrapDecryptSeed(cPassword.GetBuffer(0), cPassword.GetLength());  
CwrapDecrypt((unsigned char *) lpvBlock, 0, nFileSize);
```

Any "fix up and relocation" type services would then be performed in step 158, for example by
5 using the following command:

```
UnwrapFixUpAndRelocateDll(lpvBlock);
```

Possibly, the memory protection may be changed to execute only in step 160, for example by
10 using the VirtualProtect() command as follows:

```
DWORD lpfOldProtect; // variable to get old protection  
BOOL bVProtect = VirtualProtect((LPVOID) nPreferredLoadAddress,  
                                nPhantomFileSize,  
15                                PAGE_EXECUTE,  
                                &lpfOldProtect);
```

Function calls then can be made into that area of memory that now contains the decrypted code:

```
20 UnwrapDoDllAlgorithms();
```

Some of the "fix up" operations to be performed above include placing the addresses of external
or stub.exe functions into the address place holders of the decrypted .dll or internal code, by
using commands similar to the following:

```
25 WriteAddress((char*) 0x0a406104, (DWORD) &CallBackFunction1);  
WriteAddress((char*) 0x0a406100, (DWORD) &CallBackFunction2);
```

For instance a wrapper function could be created in the outer stub.exe that received a size
30 parameter, allocated that amount of memory off of the heap, and passed back the starting address
of that block of memory. Another example would be to have encrypted algorithms within the
hidden, encrypted .dll which would be called at run time from the front end stub once decrypted

within protected memory. The dynamic link library would be compiled and linked to expect a pointer to a function that took that parameter and/or returned a value by including prototypes in the header file as follows:

```
5 void (*lpCallbackFunc1)();
void (*lpCallbackFunc2)(unsigned long);
```

Function calls to "external" functions also could be added as follows:

```
10 (*lpCallbackFunc1)();
unsigned long z = x * x;
(*lpCallbackFunc2)(z);
```

At run time the "fix up" code would take the run time address of that "wrapper function" and place it into the pointer address within the .dll block of code as follows:

```
WriteAddress((char*) 0x0a406104, (DWORD) &CallbackFunction1);
WriteAddress((char*) 0x0a406100, (DWORD) &CallbackFunction2);
```

20 This information is readily available using the .cod output files from the compiler, an example of which follows:

```

_TestSum PROC NEAR                                ; COMDAT
; Line 8
25 00000056          push  esi
; Line 23
00001000          ff 15 00 00 00
00          call  DWORD PTR _lpCallbackFunc1
; Line 24
30 000078b4         mov  eax, DWORD PTR _a$[esp]
0000b050          push  eax
0000ce80         call  _TestSquare
```

- 23 -

```

00011      83 c4 04      add    esp, 4
00014      8b f0          mov    esi, eax
; Line 25
00016      8b 44 24 0c    mov    eax, DWORD PTR _b$[esp]
5  0001a      50            push  eax
0001b      e8 00 00 00 00 call  _TestSquare
00020      83 c4 04      add    esp, 4
00023      03 c6          add    eax, esi
; Line 28
10 00025      5e            pop    esi
00026      c3            ret    0
_TestSum ENDP
_TEXT      ENDS
;      COMDAT _TestSquare
15 _TEXT      SEGMENT
_x$ = 8
_TestSquare PROC NEAR                                ; COMDAT
; Line 30
00000      56            push  esi
20 ; Line 32
00001      8b 74 24 08    mov    esi, DWORD PTR _x$[esp]
00005      0f af f6      imul  esi, esi
; Line 34
00008      56            push  esi
25 00009      ff 15 00 00 00
      00            call  DWORD PTR _lpCallBackFunc2
0000f 83 c4 04      add    esp, 4
00012      8b c6          mov    eax, esi
; Line 36
30 00014      5e            pop    esi
00015      c3            ret    0
_TestSquare ENDP

```

Such information also is available from .map output files from the linker where the "f" between the address (i.e., 0a406100) and the object file (i.e. Algorithms.obj) means it is a "flat" address (i.e., hard coded by the linker) and the lack of an "f" means that it is an address pointer to be supplied at run time (load time) where the address that is contained in that address location is used and not the actual address location (i.e., the address that is contained at address location 5 0a406100 and not 0a406100 itself):

```

0001:00000000    _TestSum          0a401000 f Algorithms.obj
0001:00000030    _TestSquare       0a401030 f Algorithms.obj
10
0003:00001100    _lpCallbackFunc2  0a406100 Algorithms.obj
0003:00001104    _lpCallbackFunc1  0a406104 Algorithms.obj

```

When the code inside the .dll makes a "call" to a dereferenced pointer, it would jump to the correct function in the outer code and return the expected return value (if any). For example: 15

```

void CallbackFunction1(){
// This is the first function that exists in the Stub executable
// whose address has been placed at the appropriate location inside the "dll" code
20 // that has now been decrypted in a block of memory. The code inside the "dll"
// makes a function call to this function. In its encrypted state, the "dll" does not contain
// this address, but merely has a placeholder for the address. The "dll" has enough space
allocated to hold an
// address of this size. After the "dll" has been decrypted at run time, its address is
25 // placed in that location so the code inside the "dll" that references (or more
// appropriately dereferences) that address can jump (which is function call) to this
// address.
AfxMessageBox(
_T("This is the FIRST Stub.exe call back function being called from the dll.));
30     return;
}

```

- 25 -

```
void CallBackFunction2(DWORD nNumber){
// See comment for CallBackFunction1 except this function receives a parameter off
// of the stack. It could also return a value as well.
    CString
5    cString(
T("This is the SECOND Stub.exe call back function being called from the dll"));

    har buffer[20];
    ltoa(nNumber, buffer, 10);
10
    cString += _T(" with a parameter of ");
    cString += buffer;
    cString += _T(".");
    AfxMessageBox(cString.GetBuffer(0));
15    return;
}
```

The outer stub.exe would make the same kinds of jumps or function calls into the now protected decrypted code block as follows:

```
20    DWORD c;

// This command declares a function pointer. This command is different for different function
// calls. Here the called function takes two integer parameters and
25 // passes back a DWORD.
    DWORD (*lpFunc)(DWORD,DWORD);

// The function pointer is then pointed to the starting address of the function in the
// block of memory that now holds the decrypted DLL.
30 lpFunc = (DWORD (*)(DWORD,DWORD)) UnwrapFixUpAndRelocateDll();

// Now call that "function" which is really like all function calls, i.e., a jump to
```

- 26 -

```
// the address where that function exists. In this case, two
// variables are passed to that function and returning a value from that function. This function
// illustrates that the function call
// can be more complicated than merely a simple jump
5 // to an address. Inline assembler code may be used to push the variables onto
// the stack frame and return the variable from the eax register, but this function enables
// the C++ compiler to do the same function.
c = (DWORD) (*lpFunc)(a, b);
```

10 This mechanism requires the unwrap procedure and the now decrypted code to have intimate knowledge about procedural interfaces of each other but no knowledge about each other's implementation. This is the way most executable .exe files and .dll files behave but with the addition of a series of "wrapper" functions on either side for communication. This method works under Windows 95 and Windows NT 4.0 operating systems and should work under Windows NT
15 3.51 and other operating systems.

Another modified version of this mechanism that works under the Windows NT 4.0 operating system because of functions specific to Windows NT 4.0 would be to have another hidden and/or encrypted executable within the digital information product. This executable would be copied to a physical disk in an unencrypted form, launched or loaded with the
20 CreateProcess() command in its current form but called with a parameter to load the executable in suspended mode:

```
BOOL success = CreateProcess(cFrontEndExe.GetBuffer(0), 0, 0, 0, TRUE,
CREATE_NEW_CONSOLE | CREATE_SUSPENDED,
25 0, 0, &startUpInfo, &processInfo);
```

Then the first process would copy the encrypted dll into its own process and decrypt it, allocate enough memory using VirtualAllocEx() in its current form in the second process that has just loaded the expendable front end executable in a suspended state as follows:

```
30 LPVOID lpvBlockEx = VirtualAllocEx(processInfo.hProcess,
```

- 27 -

```
(LPVOID) nPreferredLoadAddress, nPhantomFileSize,
MEM_RESERVE | MEM_COMMIT,
PAGE_READWRITE);
```

- 5 The decrypted code is copied from the first process to the second suspended process using WriteProcessMemory() in its current form:

```
BOOL bWriteProcessMemory = WriteProcessMemory((HANDLE) processInfo.hProcess,
(LPVOID) lpvBlockEx, (LPVOID) nPreferredAddress,
10 (DWORD) nPhantomFileSize, (LPDWORD) &nBytesWritten);
```

The primary thread of the previously launched second process is then resumed:

```
DWORD nResumed = ResumeThread(processInfo.hThread);
15
```

Any necessary function pointers are then placed in the correct locations by the second process, the area of memory is locked to prevent any writes to a swap file, and the memory protection is changed to execute only as follows:

```
20 WriteAddress((char*) 0x0a406104, (DWORD) &CallBackFunction1);
WriteAddress((char*) 0x0a406100, (DWORD) &CallBackFunction2);
```

```
BOOL bVLock = VirtualLock((LPVOID) nPreferredLoadAddress, nPhantomFileSize);
DWORD lpfOldProtect; // variable to get old protection
25 BOOL bVProtect = VirtualProtect((LPVOID) nPreferredLoadAddress,
nPhantomFileSize, PAGE_EXECUTE, &lpfOldProtect);
```

- The program can continue running by making and receiving calls to and from the decrypted dynamic link library that now resides in the protected memory of its process using commands
- 30 such as the following:

```
DWORD c;
```

```
DWORD (*lpFunc)(DWORD,DWORD);  
lpFunc = (DWORD (*)(DWORD,DWORD)) ExpendableGetEntryAddress();  
c = (DWORD) (*lpFunc)(a, b);
```

- 5 The first process can either close down or launch another instance of that same process.

In either of these implementations using the same process or launching into a second process, the hidden encrypted code never passes through a file system driver or memory resident program in decrypted form. Code can be split up among different dynamic link libraries so that no two would reside in memory at the same time in order to protect code further. Both of these systems can be implemented using the Win32 function calls. If additional functions, similar to a
10 CreateProcess() command or a LoadLibrary() command but that take a process identifier and address location in memory to load in an executable instead of a physical file, are provided in an operating system then the entire executable and dynamic link library can be hidden, encrypted, and protected on the physical disk and then decrypted within protected memory and use the
15 operating system loader to load it directly to the operating system from memory without residing in decrypted form on any physical medium.

Having described the operation and use of the computer program product in accordance with the invention, embodiments of which are described above in connection with Figs. 3-8, and the operation of the unwrap procedure and device driver it contains, the process of constructing
20 such a computer program product will now be described in more detail. Referring now to Fig. 9, an embodiment of this process for creating a computer program product is shown. This process can be applied to any digital information including an arbitrary executable computer program, dynamic link libraries and related files of data. All digital information is treated as mere data by this process. Each separate data file is combined into a single file by this process, with an
25 executable program for performing the unwrap procedure, and optionally executable program code for a virtual device driver, into the computer program product. Each file of hidden information has a unique location and is identified by its own begin and end markers as shown in Fig. 3. The first step of this process is opening a new data file for the computer program using a name that will be used to indicate an executable file (step 200). For example, an executable
30 word processing program may be named "word_processor.exe" in the Windows95 operating system.

The three portions of the computer program product are then inserted into the open data file. First, the unwrap procedure is inserted at the beginning of the file in an executable format in step 202. The begin tag for the optional device driver is then inserted in step 204. The executable device driver program code is then inserted in step 206, followed by its corresponding end tag in step 208. For each hidden file to be inserted into this computer program product, steps 210 to 216 are performed. First, the begin tag is inserted in step 210. The begin tag also may include an indication of a unique name of the file which will be used as its name in the phantom directory created by the unwrap procedure. The hidden file is then encrypted and/or compressed in step 212 and inserted into the data file in step 214. The end tag for the hidden file is then inserted in step 216. The device driver and all of the tags may be encrypted also if the unwrap procedure has suitable decryption procedures. The computer program file is closed when the last hidden file is processed.

Using the present invention digital information, such as executable program code or various kinds of data, is loaded and unloaded as needed, and thus does not take up any more memory than is necessary. At no time does unencrypted digital information, such as computer program code, exist on disk in accessible and complete decrypted form. Because the original digital information is available as a read only file in one embodiment of the invention accessible only to the device driver, the digital information may be accessed over networks, from a CD-ROM or from a DVD, and can be made to have a limited number of uses. This mechanism is particularly useful for controlling distribution of computer programs, digitized movies or other information while reducing the cost of such distribution and control. For example, software may be distributed over a network on a single use basis, and charges may be levied on a per use basis. The ability to reverse engineer an application program also may be reduced.

One benefit with this system over some other systems for preventing unauthorized access to digital information is that the content provider maintains control of the encryption applied to the information how it may be decrypted. Any need for either a centralized facility or a predetermined decryption program is eliminated. An operating systems manufacturer or other platform vendor merely provides the capability for the information to be accessed and decrypted on the fly. Since the valuable information and any other tables of authorization codes, passwords, or hardware identifiers that the content provider may use to secure the information resides in one large encrypted file, it becomes difficult, if not impossible, for someone to determine just where any of this information exists.

A potential scenario with authorization procedure in which the present invention may be used is the following. A consumer purchases a DVD disk containing a movie. The user puts the disk into the player. This is the first time the disk is installed. The content provider's functions are loaded into the DVD chip, which looks in the encrypted table and sees that this is the first
5 time this disk is being played. The player then displays on a screen a numeric identifier and toll free phone number. The consumer calls the toll free phone number and inputs the numeric identifier that was displayed on the screen. The content provider provides a numeric password based on the numeric identifier that the user inputs into the DVD. The content provider may develop a database of information about its consumers that also may be used to detect pirating of
10 the digital information product. Now that this authorization has taken place, the software that the content provider wrote, and is now in the DVD chip, takes a hardware identifier from the DVD and encrypts it and puts it in the encrypted and buried table on the disk. Alternatively, the data may be decrypted in memory and re-encrypted back onto the disk using the hardware identifier as part of a key. Now that disk will run and show the movie and will only run on that DVD and
15 no other. The content provider could allow for a table of hardware id's so they could limit the number of DVD's that disk would run on or a limited number of times it can be shown. It should be understood that many other authorization procedures may be used.

In the foregoing scenario, the movie is encrypted on the same disk inside of the encrypted file that contains the table and functions the content provider distributed. The movie is decrypted
20 by the decryption functions contained in the file directly to the DVD chip. At no time does the movie reside anywhere in decrypted form. The content provider can protect the movie with any desired level of security (for both encryption and authorization).

In the present invention, the onus of protection of content does not reside with a hardware manufacturer or platform provider but in the hands of the content provider. The hardware
25 manufacturer only provides the mechanism to protect the digital information through the operating system. The technique and implementation of protection resides in the hands of the content provider. This mechanism allows the content providers to change the level of security as needed without any modifications to the hardware. The security of the content is provided by the encryption/decryption algorithms, public/private keys, and authorization methods which are
30 determined by the content provider. Even each individual product can have its own encryption/decryption algorithms and/or public/private keys. All of these can be changed and enhanced as the market demands.

The present invention also could be used for on-line or live use of digital information. For example, a movie could be retrieved on demand and recorded by a consumer. A set top box could receive the digital information, decrypt it, and then re-encrypt and store the information using, for example, a hardware identifier of the set top box. Since home movies digitally recorded would be encrypted using the hardware identifier of the device used in recording, that home movie could not be played on another or only on a limited number of other devices and/or for only a specified number of times depending on the wishes of the content provider. Since the algorithms are downloaded at the time of recording from a service provider, e.g., the cable company, the content provider (movie company) would provide the encrypted data to the service provider to present to their customers. The service provider need not be concerned with the encryption/decryption and authorization functions used by the content provider. Similar uses are possible with other data transmission systems including, but not limited to, telephone, cellular communications, audio transmission including communication and the like.

In another embodiment, the stub executable program is a first process that is implemented similar to a debugging tool such as the SoftIce debugger from NuMega Technologies or the WinDebug debugger from Microsoft Corporation for Ring 0 kernel level debugging for an Intel processor based architecture, or the CodeView debugger for ring 3 application level debugging. Such a debugger controls execution of a program to be debugged as a second process and steps through each program statement or opcode of the debugged program. The debugging tool could be modified to monitor each opcode that indicates a jump to a program fragment, such as each instruction or a block code. If the program fragment to be executed is not decrypted, the modified debugger decrypts the program fragment before the jump command is allowed to execute. Each program fragment may be re-encrypted after execution. Clearly, unnecessary debugging commands may be omitted from the modified debugger.

Having now described a few embodiments of the invention, it should be apparent to those skilled in the art that the foregoing is merely illustrative and not limiting, having been presented by way of example only. Numerous modifications and other embodiments are within the scope of one of ordinary skill in the art and are contemplated as falling within the scope of the invention as defined by the appended claims and equivalent thereto.

CLAIMS

1. A computer-implemented process for executing encrypted computer program logic while maintaining protection against copying of corresponding decrypted executable computer program logic, wherein the encrypted computer program logic is stored in association with first executable computer program logic, the process comprising the steps of:

5 through an operating system of a computer, reading, loading and executing the first executable computer program logic as a first process having a protected memory area defined by the operating system;

10 the first process decrypting the encrypted computer program logic into second executable computer program logic and storing the second executable computer program logic in the protected memory area; and

the first process causing loading and execution of the decrypted second computer program logic in the protected memory area.

15 2. The process of claim 1, wherein the encrypted computer program logic and the first executable computer program logic are stored in a single data file accessible through the operating system.

3. The process of claim 1, wherein the execution of the decrypted second computer program logic is performed as a second process having a second protected memory area defined by the operating system.

4. A digital information product including a computer readable medium having digital information stored thereon, the digital information including computer program logic defining first executable computer program logic, wherein the first executable computer program logic when executed performs the following steps:

25 storing the encrypted computer program logic in a data file accessible through an operating system of a computer, wherein the data file also includes first executable computer program logic;

30 through the operating system, reading, loading and executing the first executable computer program logic from the data file as a first process having a protected memory area;

the first process decrypting the encrypted computer program logic into second executable computer program logic and storing the second executable computer program logic in the protected memory area; and

the first process causing loading and execution of the decrypted second computer
5 program logic in the protected memory area.

5. A computer system comprising:

a processor for executing computer program logic;

a main memory operatively connected to the processor for storing digital information
10 including executable computer program logic at memory locations addressed by the processor;
and

an operating system defined by executable computer program logic stored in the memory
and executed by the processor and having a command which when executed by the processor
defines means for creating a process in response to a request specifying a process identifier and a
15 memory location in the main memory, wherein the process identifier indicates the process
making the request and the memory location stores executable computer program logic which
when executed defines the process.

6. A computer system having an operating system, for decrypting digital information,
20 comprising:

means for storing the encrypted computer program logic in a data file accessible through
the operating system, wherein the data file also includes first executable computer program logic;

means, invocable through the operating system, for reading, loading and executing the
first executable computer program logic from the data file as a first process having a protected
25 memory area;

the first process defining means for decrypting the encrypted computer program logic
into second executable computer program logic and storing the second executable computer
program logic in the protected memory area; and

the first process defining means for causing loading and execution of the decrypted
30 second computer program logic in the protected memory area.

7. The computer system of claim 6, wherein the encrypted computer program logic and the first executable computer program logic are stored in a single data file accessible through the operating system.
- 5 8. The computer system of claim 6, wherein the execution of the decrypted second computer program logic is performed as a second process having a second protected memory area defined by the operating system.
9. A digital information product, including a computer readable medium with computer readable
10 information stored thereon, wherein the computer readable information comprises:
a first portion of executable computer program logic; and
a second portion of encrypted digital information; and
wherein the first portion of executable program logic, when executed, defines means,
operative in response to requests for digital information, for accessing the second portion of
15 encrypted digital information, for decrypting the encrypted digital information, and for
outputting the decrypted digital information.
10. The digital information product of claim 9, wherein the encrypted digital information is
encrypted executable computer program logic.
- 20 11. A computer program product including a self-decrypting encrypted executable computer program, comprising:
a computer readable medium having computer program logic stored thereon, wherein the
computer program logic defines:
25 a first module,
a second module,
wherein the first module, when executed by a computer, defines means for loading the
second module into memory of the computer, and
a third module defining the encrypted executable computer program,
30 wherein the second module, when executed by a computer, defines means for
communicating with an operating system of the computer to receive requests for program code
from the encrypted executable computer program from the third module, and for processing the

requests to access and decrypt the encrypted executable computer program and for providing the decrypted executable code from the third module to the operating system.

12. A process for executing encrypted executable computer programs on a computer system
5 having a processor, memory and operating system, comprising the steps of:
- receiving computer program logic having a first module defining a start up routine, a
second module, and a third module containing the encrypted executable computer program;
 - executing the first module of the received computer program logic using the processor,
wherein the step of executing causes the second module to be loaded into the memory of
10 the computer system, and
 - generating requests from the operating system for data from the encrypted executable
computer program which are received by the second module, and
 - accessing and decrypting the encrypted executable computer program and returning the
decrypted executable computer program to the operating system.

15

1/8

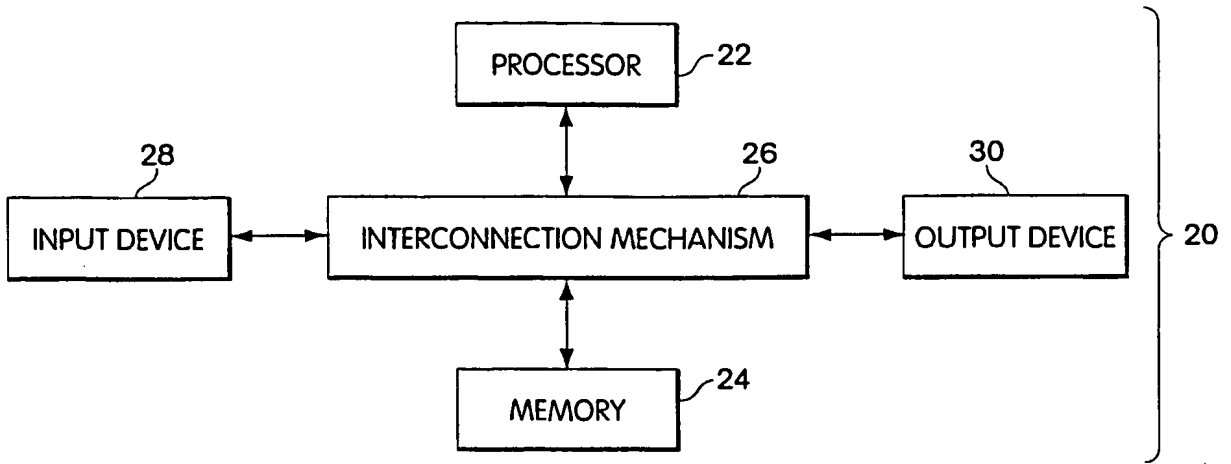


Fig. 1

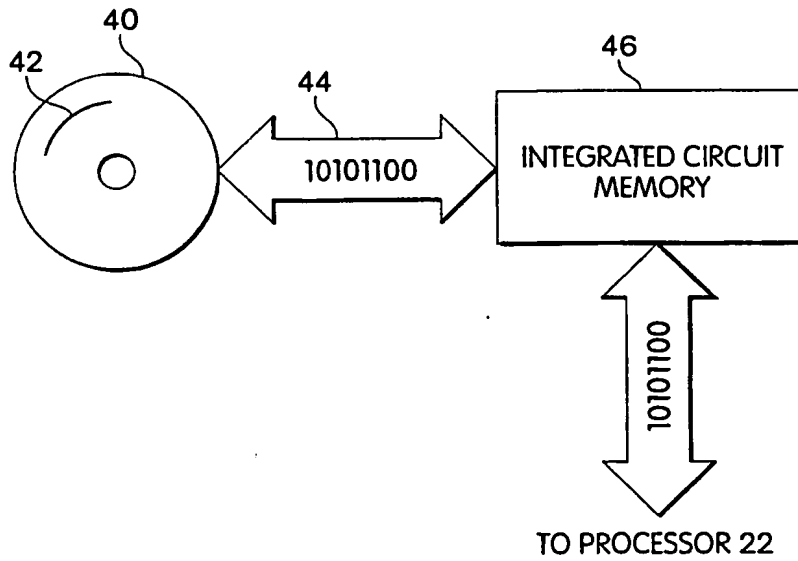


Fig. 2

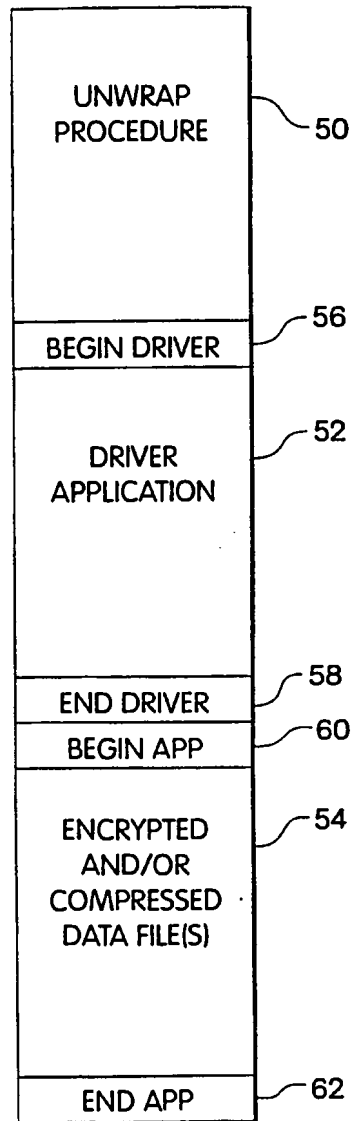


Fig. 3

3/8

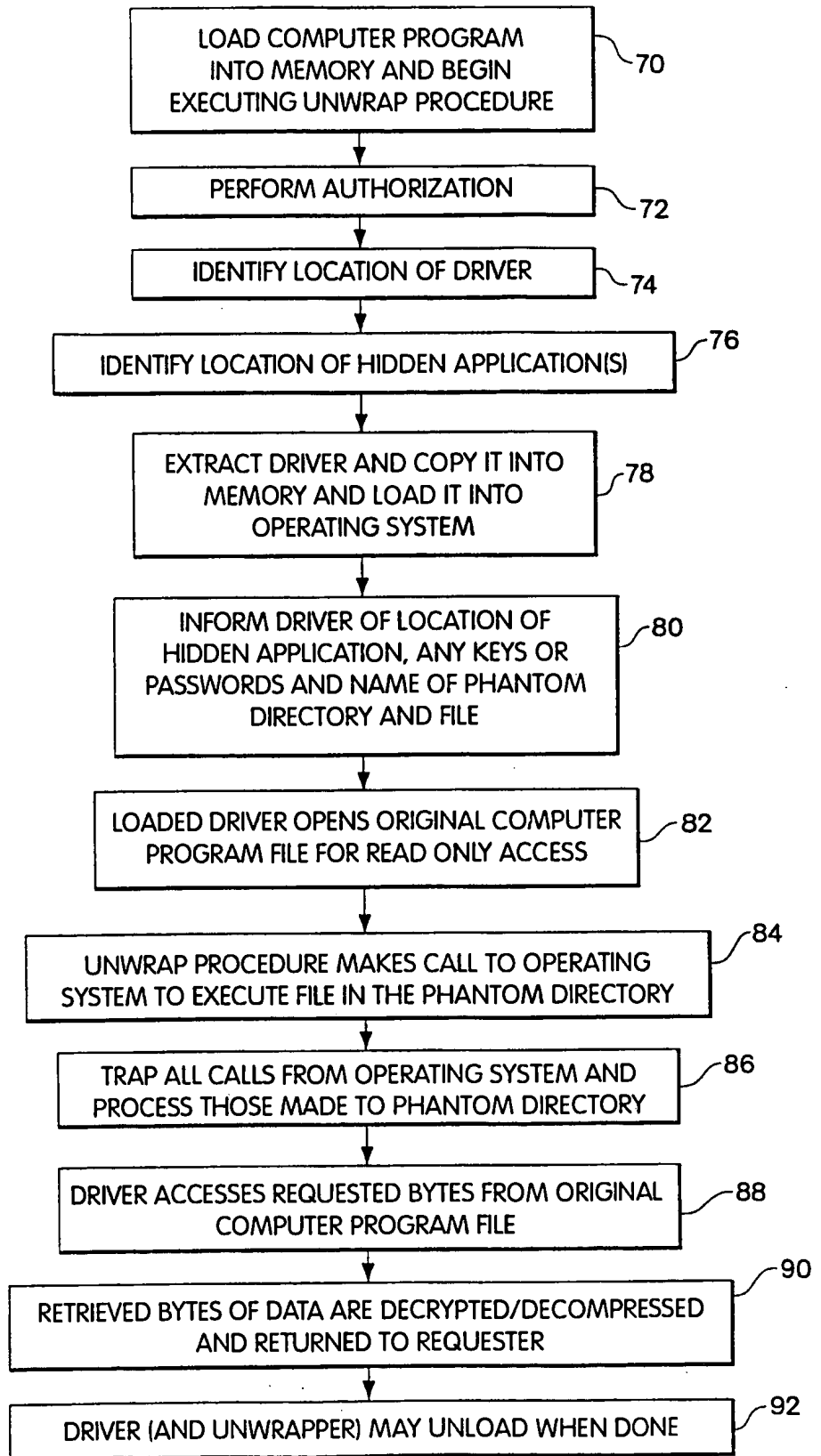


Fig.4

SUBSTITUTE SHEET (RULE 26)

4/8

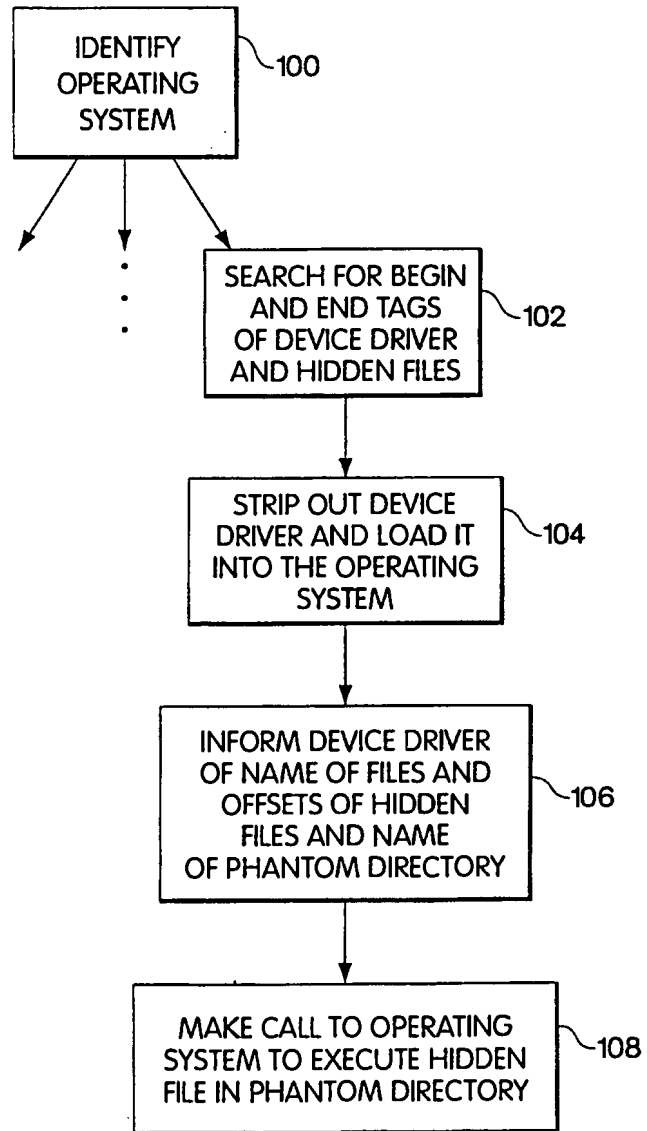


Fig. 5

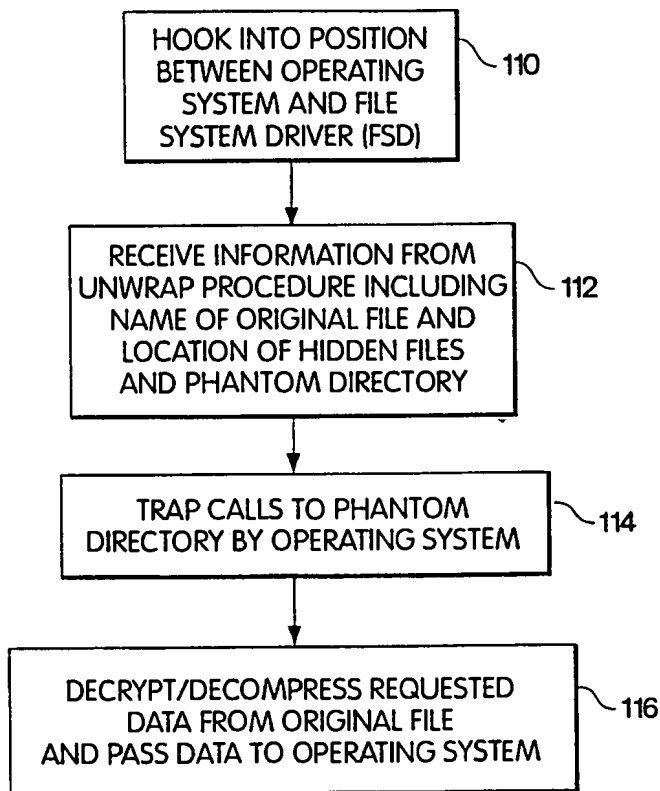


Fig. 6

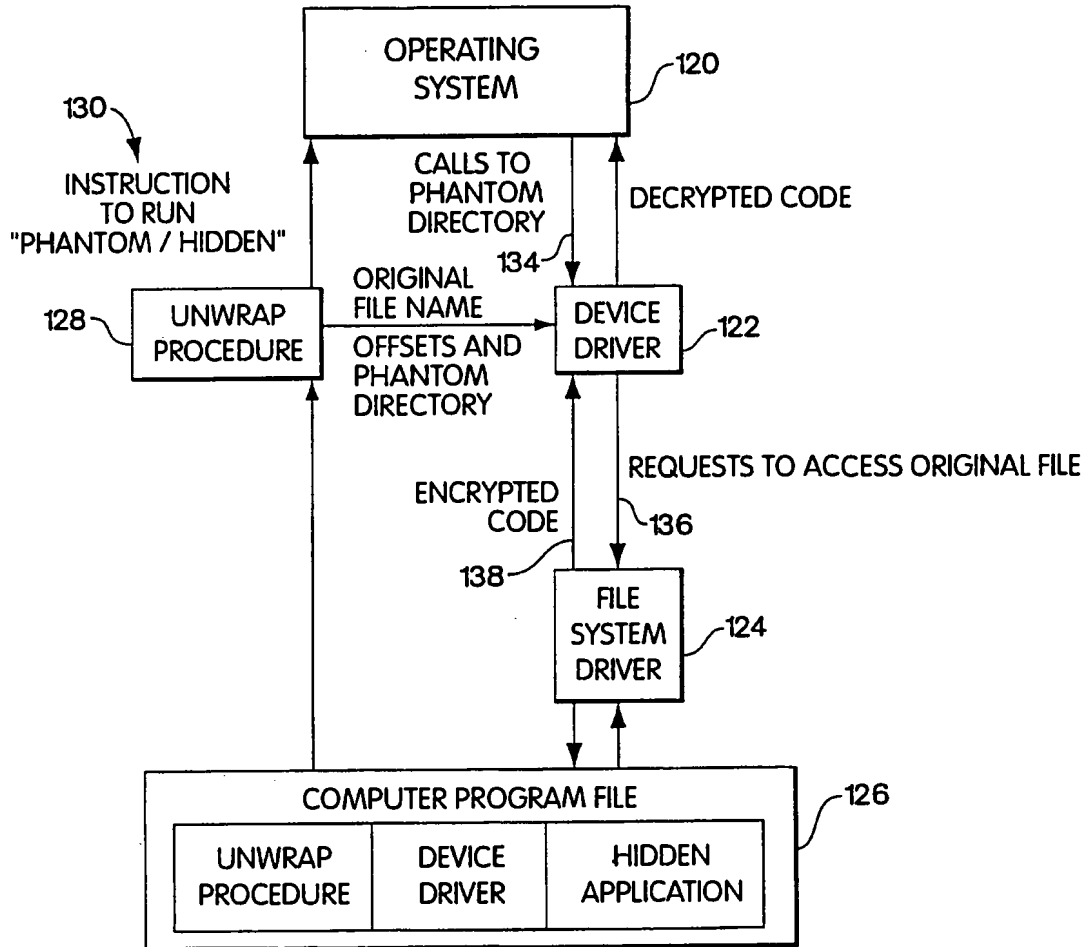


Fig. 7

7/8

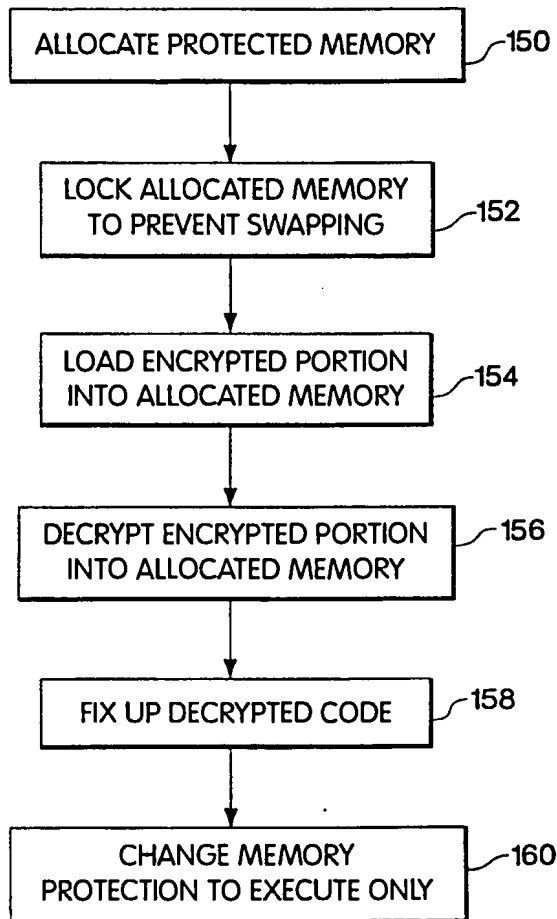


Fig. 8

8/8

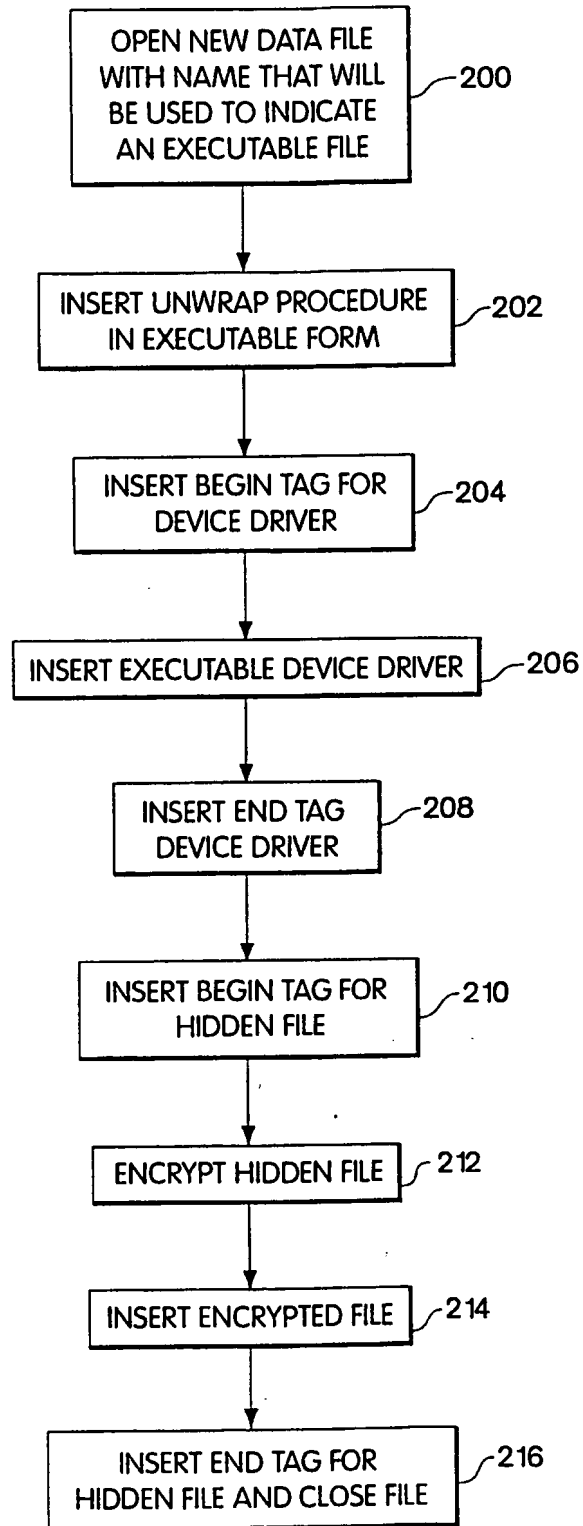


Fig. 9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US97/16223

A. CLASSIFICATION OF SUBJECT MATTER IPC(6) :H04L 9/00 US CL : 380/4 According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) U.S. : 380/4,9,23,25,49,50,59 Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 4,937,861 A (CUMMINS) 26 June 1990, see Abstract.	1-12
A	US 5,007,082 A (CUMMINS) 09 April 1991, see Abstract.	1-12
A	US 5,144,659 A (JONES) 01 September 1992, see Abstract.	1-12
A	US 5,155,827 A (GHERING) 13 October 1992, see Abstract.	1-12
A	US 5,396,609 A (SCHMIDT et al) 07 March 1995, see Abstract.	1-12
<input type="checkbox"/> Further documents are listed in the continuation of Box C. <input type="checkbox"/> See patent family annex.		
* Special categories of cited documents:		* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
A document defining the general state of the art which is not considered to be of particular relevance		*X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
B earlier document published on or after the international filing date		*Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
L document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)		*A* document member of the same patent family
O document referring to an oral disclosure, use, exhibition or other means		
P document published prior to the international filing date but later than the priority date claimed		
Date of the actual completion of the international search 20 JANUARY 1998	Date of mailing of the international search report 18 FEB 1998	
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer <i>Diane Goodenough</i> BERNARR EARL GREGORY Telephone No. (703) 306-4153	

③ **EUROPEAN PATENT SPECIFICATION**

- ④ Date of publication of patent specification: 18.04.90 ⑤ Int. Cl.⁵: G 06 F 9/46
⑥ Application number: 82302596.0
⑦ Date of filing: 21.05.82
⑧ Divisional applications 88200917, 88200916,
88200921 filed on 09.05.88.

⑨ **Digital data processing system.**

⑩ Priority: 22.05.81 US 266413
22.05.81 US 266539 22.05.81 US 266414
22.05.81 US 266521 22.05.81 US 266532
22.05.81 US 266416 22.05.81 US 266403
22.05.81 US 266409 22.05.81 US 266408
22.05.81 US 266424 22.05.81 US 266401
22.05.81 US 266421 22.05.81 US 266524
22.05.81 US 266404

⑪ Date of publication of application:
22.12.82 Bulletin 82/51
⑫ Publication of the grant of the patent:
18.04.90 Bulletin 90/16
⑬ Designated Contracting States:
AT BE CH DE FR GB IT LI LU NL SE

⑭ References cited:
Hasselmeier, Spruth: "Rechnerstrukturen",
1974, pp 75-103
Klar, Wichmann: "Mikroprogrammierung", June
1975, pp 159-163, 176-179, 185-187, 195-205,
214-215

⑮ Proprietor: **DATA GENERAL CORPORATION**
Route 9
Westboro Massachusetts 01581 (US)

⑯ Inventor: Ahlstrom, John K.
1309 San Domar
Mountain View California 94043 (US)
Inventor: Bachman, Brett L.
214 W. Canton Street Suite 4
Boston Massachusetts 02116 (US)
Inventor: Belgard, Richard A.
21250 Glenmont Drive
Saratoga California 95070 (US)
Inventor: Bernstein, David H.
41 Bay Colony Drive
Ashland Massachusetts 01721 (US)
Inventor: Bratt, Richard Glenn
9 Brook Trail Road
Wayland Massachusetts 01778 (US)
Inventor: Clancy, Gerald F.
13069 Jaccaranda Center
Saratoga California 95070 (US)
Inventor: Farber, David A.
1700 Lakewood Avenue
Durham North Carolina 27707 (US)
Inventor: Gavrin, Edward S.
Beaver Pond Road RFD 4
Lincoln Massachusetts 01773 (US)

Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European patent convention).

Courier Press, Leamington Spa, England.

EP 0 067 556 B1

References cited:

IBM TECHNICAL DISCLOSURE BULLETIN, vol. 22, no. 3, august 1979, pages 1286-1289, New York, US D.B. LOMET: "Regions for controlling the propagation of addressability in capability systems"

IBM TECHNICAL DISCLOSURE BULLETIN, vol. 15, no. 9, February 1973, pages 2721-2722 W.L. DUNNE: "Common Compiler/Interpreter for a programming language"

ADVANCES IN COMPUTERS, vol. 14, 1976, Academic Press, New York, US pages 231-272 D.K. HSIAO et al.: "Information Secure Systems"

IEEE DIGEST OF PAPERS FROM COMPCON FALL MEETING, September 10-12, 1974, Washington, Micros and Minis, pages 15-17 Long Beach, US C.J. NEUHAUSER et al.: "Description of an emulation laboratory"

7th Annual Symposium on COMPUTER ARCHITECTURE, May 6-8, 1980, Conference Proceedings, pages 245-252 New York, US V. BERTIS: "Security and protection of data in the IBM System/38"

VERY LARGE DATA BASES, vol.9, no. 2C, October 1977, Third International Conference Proceedings, pages 507-514 Tokyo, JP D. DOWNS et al.: "A kernel design for a secure data base management system"

Inventor: Gruner, Ronald Hans
112 Dublin Wood Drive
Cary North Carolina 27514 (US)
Inventor: Houseman, David L.
1213 Selwyn Lane
Cary North Carolina 27511 (US)
Inventor: Jones, Thomas M. Jones
300 Reade Road
Chapel Hill North Carolina 27514 (US)
Inventor: Katz, Lawrence H.
10943 S. Forest Ridge Road
Oregon City Oregon 97045 (US)
Inventor: Mundie, Craig James
136 Castewood Drive
Cary North Carolina (US)
Inventor: Pilat, John F.
1308 Ravenhurst Drive
Raleigh North Carolina 27609 (US)
Inventor: Richmond, Michael S.
Fearington Post Box 51
Pittsboro North Carolina 27312 (US)
Inventor: Schleimer, Stephen I.
1208 Ellen Place
Chapel Hill North Carolina 27514 (US)
Inventor: Wallach, Steven J.
12436 Green Meadow Lane
Saratoga California 95070 (US)
Inventor: Wallach, Walter A., Jr.
1336 Medfield Road
Raleigh North Carolina 27607 (US)
Inventor: Wells, Douglas M.
106 Robin Road
Chapel Hill North Carolina 27514 (US)

Representative: Pears, David Ashley et al
REDDIE & GROSE 16 Theobalds Road
London WC1X 8PL (GB)

Description

The present invention relates to a digital data processing system and, more particularly, to a multiprocessor digital data processing system suitable for use in a data processing network and having a simplified, flexible user interface and flexible, multileveled internal mechanism.

A general trend in the development of data processing systems has been towards systems suitable for use in interconnected data processing networks. Another trend has been towards data processing systems wherein the internal structure of the system is flexible, protected from users, and effectively invisible to the user and wherein the user is presented with a flexible and simplified interface to the system.

Certain problems and shortcomings affecting the realization of such a data processing system have appeared repeatedly in the prior art and must be overcome to create a data processing system having the above attributes. These prior art problems and limitations include the following topics.

First, the data processing systems of the prior art have not provided a system wide addressing system suitable for use in common by a large number of data processing systems interconnected into a network. Addressing systems of the prior art have not provided sufficiently large address spaces and have not allowed information to be permanently and uniquely identified. Prior addressing systems have not made provisions for information to be located and identified as to type or format, and have not provided sufficient granularity. In addition, prior addressing systems have reflected the physical structure of particular data processing systems. That is, the addressing systems have been dependent upon whether a particular computer was, for example, an 8, 16, 32, 64 or 128 bit machine. Since prior data processing systems have incorporated addressing mechanisms wherein the actual physical structure of the processing system is apparent to the user, the operations a user could perform have been limited by the addressing mechanisms. In addition, prior processor systems have operated as fixed word length machines, further limiting user operations.

Prior data processing systems have not provided effective protection mechanisms preventing one user from effecting another user's data and programs without permission. Such protection mechanisms have not allowed unique, positive identification of users requesting access to information, or of information, nor have such mechanisms been sufficiently flexible in operation. In addition, access rights have pertained to the users rather than to the information, so that control of access rights has been difficult. Finally, prior art protection mechanisms have allowed the use of "Trojan Horse arguments". That is, users not having access rights to certain information have been able to gain access to that information through another user or procedure having such access rights.

Yet another problem of the prior art is that of providing a simple and flexible user's interface to a data processing system. The character of user's interface to a data processing system is determined, in part, by the means by which a user refers to and identifies operands and procedures of the user's programs and by the instruction structure of the system. Operands and procedures are customarily referred to and identified by some form of logical address having points of reference, and validity, only within a user's program. These addresses must be translated into logical and physical addresses within a data processing system each time a program is executed, and must then be frequently retranslated or generated during execution of a program. In addition, a user must provide specific instructions as to data format and handling. As such reference to operands or procedures typically comprise a major portion of the instruction stream of the user's program and requires numerous machine translations and operations to implement. A user's interface to a conventional system is thereby complicated, and the speed of execution of programs reduced, because of the complexity of the program references to operands and procedures.

A data processing system's instruction structure includes both the instructions for controlling system operations and the means by which these instructions are executed. Conventional data processing systems are designed to efficiently execute instructions in one or two user languages, for example, FORTRAN or COBOL. Programs written in any other language are not efficiently executable. In addition, a user is often faced with difficult programming problems when using any high level language other than the particular one or two languages that a particular conventional system is designed to utilize.

Yet another problem in conventional data processing systems is that of protecting the system's internal mechanisms, for example, stack mechanisms and internal control mechanisms, from accidental or malicious interference by a user.

Finally, the internal structure and operation of prior art data processing systems have not been flexible, or adaptive, in structure and operation. That is, the internal structure and operation of prior systems have not allowed the systems to be easily modified or adapted to meet particular data processing requirements. Such modifications may include changes in internal memory capacity, such as the addition or deletion of special purpose subsystems, for example, floating point or array processors. In addition, such modifications have significantly effected the users interface with the system. Ideally, the actual physical structure and operation of the data processing system should not be apparent at the user interface.

It has already been proposed (IBM Technical Disclosure Bulletin Vol. 22 No. 3 Aug. 1979 pp 1286—1289) to maintain such a large address space that every object which is ever created can have a unique identifier. This requires a very large identifier field, e.g. 40 to 50 bits.

The object of the present invention is to implement such a concept so that it may be applied across many computers geographically distributed and without requiring all computers to use the same

programming language.

The system according to the invention is defined in the appended claims.

It is known in virtual address machines to use name tables to provide logical addresses for translation to physical addresses (Klar, Wichmann, "Mikroprogrammierung" June 1975, especially pp. 159—163, 176—179, 185—187, 195—205, 214—215) and this reference also discusses the emulation in software of different target machines.

More specifically, the embodiment of the invention described in detail below provides a data processing system suitable for use in interconnected data processing networks, which internal structure is flexible, protected from users, effectively invisible to users, and provides a flexible and simplified interface to users. The data processing system provides an addressing mechanism allowing permanent and unique identification of all information generated for use in or by operation of the system, and an extremely large address space which is accessible to and common to all such data processing systems. The addressing mechanism provides addresses which are independent of the physical configuration of the system and allow information to be completely identified, with a single address, to the bit granular level and with regard to information type or format. The present invention further provides a protection mechanism wherein variable access rights are associated with individual bodies of information. Information, and users requesting access to information, are uniquely identified through the system addressing mechanism. The protection mechanism also prevents use of Trojan Horse arguments. And, the present invention provides an instruction structure wherein high level user language instructions are transformed into dialect coded, uniform, intermediate level instructions to provide equal facility of execution for a plurality of user languages. Another feature is the provision of an operand reference mechanism wherein operands are referred to in user's programs by uniform format names which are transformed, by an internal mechanism transparent to the user, into addresses. The present invention additionally provides multilevel control and stack mechanisms protecting the system's internal mechanism from interference by users. Yet another feature is a data processing system having a flexible internal structure capable of performing multiple, concurrent operations and comprised of a plurality of separate, independent processors. Each such independent processor has a separate microinstruction control and at least one separate and independent port to a central communications and memory node. The communications and memory node is also an independent processor having separate and independent microinstruction control. The memory processor is internally comprised of a plurality of independently operating, microinstruction controlled processors capable of performing multiple, concurrent memory and communications operations. The present invention also provides further data processing system structural and operational features for implementing the above features.

It is thus advantageous to incorporate the present invention into a data processing system because the present invention provides addressing mechanisms suitable for use in large interconnected data processing networks. Additionally, the present invention is advantageous in that it provides an information protection mechanism suitable for use in large, interconnected data processing networks. The present invention is further advantageous in that it provides a simplified, flexible, and more efficient interface to a data processing system. The present invention is yet further advantageous in that it provides a data processing system which is equally efficient with any user level language by providing a mechanism for referring to operands in user programs by uniform format names and instruction structure incorporating dialect coded, uniform format intermediate level instructions. Additionally, the present invention protects data processing system internal mechanisms from user interference by providing multilevel control and stack mechanisms. The present invention is yet further advantageous in providing a flexible internal system structure capable of performing multiple, concurrent operations, comprising a plurality of separate, independent processors, each having a separate microinstruction control and at least one separate and independent port to a central, independent communications and memory processor comprised of a plurality of independent processors capable of performing multiple, concurrent memory and communications operations.

Other advantages and features of the present invention will be understood by those of ordinary skill in the art, after referring to the following detailed description of the preferred embodiments and drawings wherein.

BRIEF DESCRIPTION OF DRAWINGS

- Fig. 1 is a partial block diagram of a computer system incorporating the present invention;
- Fig. 2 is a diagram illustrating computer system addressing structure of the present invention;
- Fig. 3 is a diagram illustrating the computer system instruction stream of the present invention;
- Fig. 4 is a diagram illustrating the control structure of a conventional computer system;
- Fig. 4A is a diagram illustrating the control structure of a computer system incorporating the present invention;
- Fig. 5 — Fig. A1 inclusive are diagrams all relating to the present invention;
- Fig. 5 is a diagram illustrating a stack mechanism;
- Fig. 6 is a diagram illustrating procedures, procedure objects, processes, and virtual processors;
- Fig. 7 is a diagram illustrating operating levels and mechanisms of the present computer;
- Fig. 8 is a diagram illustrating a physical implementation of processes and virtual processors;

EP 0 067 556 B1

- Fig. 9 is a diagram illustrating a process and process stack objects;
Fig. 10 is a diagram illustrating operation of macrostacks and secure stacks;
Fig. 11 is a diagram illustrating detailed structure of a stack;
Fig. 12 is a diagram illustrating a physical descriptor;
5 Fig. 13 is a diagram illustrating the relationship between logical pages and frames in a memory storage space;
Fig. 14 is a diagram illustrating access control to objects;
Fig. 15 is a diagram illustrating virtual processors and virtual processor swapping;
Fig. 16 is a partial block diagram of an I/O system of the present computer system;
10 Fig. 17 is a diagram illustrating operation of a ring grant generator;
Fig. 18 is a partial block diagram of a memory system;
Fig. 19 is a partial block diagram of a fetch unit of the present computer system;
Fig. 20 is a partial block diagram of an execute unit of the present computer system;
Fig. 101 is a more detailed partial block diagram of the present computer system;
15 Fig. 102 is a diagram illustrating certain information structures and mechanisms of the present computer system;
Fig. 103 is a diagram illustrating process structures;
Fig. 104 is a diagram illustrating a macrostack structure;
Fig. 105 is a diagram illustrating a secure stack structure;
20 Figs. 106 A, B, and C are diagrams illustrating the addressing structure of the present computer system;
Fig. 107 is a diagram illustrating addressing mechanisms of the present computer system;
Fig. 108 is a diagram illustrating a name table entry;
Fig. 109 is a diagram illustrating protection mechanisms of the present computer system;
25 Fig. 110 is a diagram illustrating instruction and microinstruction mechanism of the present computer system;
Fig. 201 is a detailed block diagram of a memory system;
Fig. 202 is a detailed block diagram of a fetch unit;
Fig. 203 is a detailed block diagram of an execute unit;
30 Fig. 204 is a detailed block diagram of an I/O system;
Fig. 205 is a partial block diagram of a diagnostic processor system;
Fig. 206 is a diagram illustrating assembly of Figs. 201—205 to form a detailed block diagram of the present computer system;
Fig. 207 is a detailed block diagram of a memory interface controller;
35 Fig. 209 is a diagram of a memory to I/O system port interface;
Fig. 210 is a diagram of a memory operand port interface;
Fig. 211 is a diagram of a memory instruction port interface;
Fig. 230 is a detailed block diagram of memory field interface unit logic;
Fig. 231 is a diagram illustrating memory format manipulation operations;
40 Fig. 238 is a detailed block diagram of fetch unit offset multiplexer;
Fig. 239 is a detailed block diagram of fetch unit bias logic;
Fig. 240 is a detailed block diagram of a generalized four way, set associative cache representing name cache, protection cache, and address translation unit;
Fig. 241 is a detailed block diagram of portions of computer system instruction and microinstruction control logic;
45 Fig. 242 is a detailed block diagram of portions of computer system microinstruction control logic;
Fig. 243 is a detailed block diagram of further portions of computer system microinstruction control logic;
Fig. 244 is a diagram illustrating computer system states of operation;
50 Fig. 245 is a diagram illustrating computer system states of operation for a trace trap request;
Fig. 246 is a diagram illustrating computer system states of operation for a memory repeat interrupt;
Fig. 247 is a diagram illustrating priority level and masking of computer system events;
Fig. 248 is a detailed block diagram of event logic.
Fig. 249 is a detailed block diagram of microinstruction control store logic;
55 Fig. 251 is a diagram illustrating a return control word stack word;
Fig. 252 is a diagram illustrating machine control words;
Fig. 253 is a detailed block diagram of a register address generator;
Fig. 254 is a block diagram of interval and egg timers;
Fig. 255 is a detailed block diagram of execute unit control logic;
60 Fig. 257 is a detailed block diagram of execute unit multiplier data paths and memory;
Fig. 260 is a diagram illustrating operation of an execute unit command queue load and interface to a fetch unit;
Fig. 261 is a diagram illustrating operation of an execute unit operand buffer load and interface to a fetch unit;
65 Fig. 262 is a diagram illustrating operation of an execute unit storeback or transfer of results and

interface to a fetch unit;

Fig. 263 is a diagram illustrating operation of an execute unit check test condition and interface to a fetch unit;

5 Fig. 264 is a diagram illustrating operation of an execute unit exception test and interface to a fetch unit;

Fig. 265 is a block diagram of an execute unit arithmetic operation stack mechanism;

Fig. 266 is a diagram illustrating execute unit and fetch unit interrupt handshaking and interface;

Fig. 267 is a diagram illustrating execute unit and fetch unit interface and operation for nested
10 interrupts;

Fig. 268 is a diagram illustrating execute unit and fetch unit interface and operation for loading an execute unit control store;

Fig. 269 is a detailed block diagram and illustration of operation of an I/O system ring grant generator;

Fig. 270 is a detailed block diagram of a fetch unit micromachine of the present computer system;

Fig. 271 is a diagram illustrating a logical descriptor;

15 Fig. 272 is a diagram illustrating use of fetch unit stack registers;

Fig. 273 is a diagram illustrating structures controlling event invocations;

Fig. 301 is a diagram illustrating pointer formats;

Fig. 302 is a diagram illustrating an associated address table;

Fig. 303 is a diagram illustrating a namespace overview of a procedure object;

20 Fig. 304 is a diagram illustrating name table entries;

Fig. 305 is a diagram illustrating an example of name resolution;

Fig. 306 is a diagram illustrating name cache entries;

Fig. 307 is a diagram illustrating translation of S-interpreter universal identifiers to dialect numbers;

Fig. 401 is a diagram illustrating operating systems and system resources;

25 Fig. 402 is a diagram illustrating multiprocess operating systems;

Fig. 403 is a diagram illustrating an extended operating system and a kernel operating system;

Fig. 404 is a diagram illustrating an EOS view of objects;

Fig. 405 is a diagram illustrating pathnames to universal identifier translation;

Fig. 406 is a diagram illustrating universal identifier detail;

30 Fig. 407 is a diagram illustrating address translation with an address translation unit, a memory hash table, and a memory;

Fig. 408 is a diagram illustrating hashing in an active subject table;

Fig. 409 is a diagram illustrating logical allocation units and objects;

Fig. 410 is a diagram illustrating an active logical allocation unit table and active allocation units;

35 Fig. 411 is a diagram illustrating a conceptual logical allocation unit directory structure;

Fig. 412 is a diagram illustrating detail of a logical allocation unit directory entry;

Fig. 413 is a diagram illustrating universal identifiers and active object numbers;

Fig. 416 is a diagram illustrating subject templates, primitive access control list entries, and extended
40 access control list entries;

Fig. 421 is a diagram illustrating an active primitive access matrix and an active primitive access matrix
entry;

Fig. 422 is a diagram illustrating primitive data access checking;

Fig. 448 is a diagram illustrating event counters and await entries;

Fig. 449 is a diagram illustrating an await table overview;

45 Fig. 453 is a diagram illustrating an overview of a virtual processor;

Fig. 454 is a diagram illustrating virtual processor synchronization;

Fig. 467 is a diagram illustrating an overview of a macrostack object;

Fig. 468 is a diagram illustrating details of a macrostack object base;

Fig. 469 is a diagram illustrating details of a macrostack frame;

50 Fig. 470 is a diagram illustrating an overview of a secure stack;

Fig. 471 is a diagram illustrating details of a secure stack frame; and,

Fig. 472 is a diagram illustrating an overview of procedure object.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

55 The following description presents the structure and operation of a computer system incorporating a presently preferred embodiment of the present invention. As indicated in the following Table of Contents, certain features of computer system structure and operation will first be described in an Introductory Overview. Next, these and other features will be described in further detail in a more detailed Introduction to the detailed descriptions of the computer system. Following the Introduction, the structure and
60 operation of the computer system will be described in detail. The detailed descriptions will present descriptions of the structure and operation of each of the major subsystems, or elements, of the computer system, of the interfaces between these major subsystems, and of overall computer system operation. Next, certain features of the operation of the individual subsystems will be presented in further detail.

65 Certain conventions are used throughout the following descriptions to enhance clarity of presentation. First, and with exception of the Introductory Overview, each figure referred to in the following descriptions

will be referred to by a three digit number. The most significant digit represents the number of the chapter in the following descriptions in which a particular figure is first referred to. The two least significant digits represent the sequential number of appearance of a figure in a particular chapter. For example, Figure 319 would be the nineteenth figure appearing in the third chapter. Figures appearing in the Introductory Overview are referred to by a one or two digit number representing the order in which they are referred to in the Introductory Overview. It should be noted that certain figure numbers, for example, Figure 208, do not appear in the following figures and descriptions; the subject matter of these figures has been incorporated into other figures and these figures deleted, during drafting of the following descriptions, to enhance clarity of presentation.

Second, reference numerals comprise a two digit number (00—99) preceded by the number of the figure in which the corresponding elements first appear. For example, reference numerals 31901 to 31999 would refer to elements 1 through 99 appearing in Fig. 319.

Finally, interconnections between related circuitry is represented in two ways. First, to enhance clarity of presentation, interconnections between circuitry may be represented by common signal names or references, rather than by drawn representations of wires or buses. Second, where related circuitry is shown in two or more figures, the figures may share a common figure number and will be distinguished by a letter designation, for example, Figs. 319, 319A, and 319B. Common electrical points between such circuitry may be indicated by a bracket enclosing a lead to such a point and a designation of the form "A—b". "A" indicates other figures having the same common point for example, 319A, and "b" designates the particular common electrical point. In cases of related circuitry shown in this manner in two or more figures, reference numerals to elements will be assigned in sequence through the group of figures; the figure number portion of such reference numerals will be that of the first figure of the group of figures.

INTRODUCTORY OVERVIEW

A. Hardware Overview (Fig. 1)

B. Individual Operating Features (Figs. 2, 3, 4, 5, 6)

- 1. Addressing (Fig. 2)
- 2. S-Language Instructions and Namespace Addressing (Fig. 3)
- 3. Architectural Base Pointer Addressing
- 4. Stack Mechanisms (Figs. 4-5)

C. Procedure Processes and Virtual Processors (Fig. 6)

D. CS 101 Overall Structure and Operation (Figs. 7, 8, 9, 10, 11, 12, 13, 14, 15)

- 1. Introduction (Fig. 7)
- 2. Compilers 702 (Fig. 7)
- 3. Binder 703 (Fig. 7)
- 4. EOS 704 (Fig. 7)
- 5. KOS and Architectural Interface 708 (Fig. 7)
- 6. Processes 610 and Virtual Processors 612 (Fig. 8)
- 7. Processes 610 and Stacks (Fig. 9)
- 8. Processes 610 and Calls (Figs. 10, 11)
- 9. Memory References and the Virtual Memory Management System (Fig. 12, 13)
- 10. Access Control (Fig. 14)
- 11. Virtual Processors and Virtual Processor Swapping (Fig. 15)

E. CS 101 Structural Implementation (Figs. 16, 17, 18, 19, 20)

- 1. (IOS) 116 (Figs. 16, 17)
- 2. Memory (MEM) 112 (Fig. 18)
- 3. Fetch Unit (FU) 120 (Fig. 19)
- 4. Execute Unit (EU) 122 (Fig. 20)

1. Introduction (Figs. 101—110)

A. General Structure and Operation (Fig. 101)

- a. General Structure
- b. General Operation
- c. Definition of Certain Terms
- d. Multi-program Operation
- e. Multi-Language Operation
- f. Addressing Structure
- g. Protection Mechanism

B. Computer System 10110 Information Structure and Mechanisms (Figs. 102, 103, 104, 105)

- a. Introduction (Fig. 102)
- b. Process Structures 10210 (Figs. 103, 104, 105)
 - 1. Procedure Objects (Fig. 103)
 - 2. Stack Mechanisms (Figs. 104, 105)
 - 3. FURSM 10214 (Fig. 103)

C. Virtual Processor State Blocks and Virtual Process Creation (Fig. 102)

EP 0 067 556 B1

- D. Addressing Structures 10220 (Figs. 103, 106, 107, 108)
 - 1. Objects, UID's, AON's, Names, and Physical Addresses (Fig. 106)
 - 2. Addressing Mechanisms 10220 (Fig. 107)
 - 3. Name Resolution (Figs. 103, 108)
 - 5 4. Evaluation of AON Addresses to Physical Addresses (Fig. 107)
 - E. CS 10110 Protection Mechanisms (Fig. 109)
 - F. CS 10110 Micro-Instruction Mechanisms (Fig. 110)
 - G. Summary of Certain CS 10110 Features and Alternate Embodiments.
10. 2. Detailed Description of CS 10110 Major Subsystems Figs. 201—206, 207—274
- A. MEM 10110 (Figs. 201, 206, 207-237)
 - a. Terminology
 - b. MEM 10112 physical Structure (Fig. 201)
 - c. MEM 10112 General Operation
 - 15 d. MEM 10112 Port Structure
 - 1. IO Port Characteristics
 - 2. JO Port Characteristics
 - 3. JI Port Characteristics
 - e. MEM 10112 Control Structure and Operation (Fig. 207)
 - 20 1. MEM 10112 Control Structure
 - 2. MEM 10112 Control Operation
 - f. MEM 10112 Operations
 - g. MEM 10112 Interfaces to JP 10114 and IOS 10116 (Figs. 209, 210, 211, 204)
 - 25 1. IO Port 20910 Operating Characteristics (Figs 209, 204)
 - 2. JO Port 21010 Operating Characteristics (Fig. 210)
 - 3. JI Port 21110 Operating Characteristics (Fig. 211)
 - h. FIU 20120 (Figs. 201, 230, 231)
 - B. Fetch Unit 10120 (Figs. 202, 206, 101, 103, 104, 238)
 - 30 1. Descriptor Processor 20210 (Figs. 202, 101, 103, 104, 238, 239).
 - a. Offset Processor 20218 Structure
 - b. AON Processor 20216 Structure
 - c. Length Processor 20220 Structure
 - d. Descriptor Processor 20218 Operation
 - a.a. Offset Selector 20238
 - 35 b.b. Offset Multiplexer 20240 Detailed Structure (Fig. 238)
 - c.c. Offset Multiplexer 20240 Detailed Operation
 - aaa. Internal Operation
 - bbb. Operation Relative to DESP 20210
 - e. Length Processor 20220 (Fig. 239)
 - 40 a.a. Length ALU 20252
 - b.b. BIAS 20246 (Fig. 239)
 - f. AON Processor 20216
 - a.a. AONGRF 20232
 - b.b. AON Selector 20248
 - 45 2. Memory Interface 20212 (Figs. 106, 240)
 - a.a. Descriptor Trap 20256 and Data Trap 20258
 - b.b. Name Cache 10226, Address Translation Unit 10228, and Protection Cache 10234 (Fig. 106)
 - c.c. Structure and Operation of Generalized Cache and NC 10226 (Fig. 240)
 - d.d. ATU 10228 and PC 10234
 - 50 3. Fetch Unit Control Logic 20214 (Fig. 202)
 - a.a. Fetch Unit Control Logic 20214 Overall Structure
 - b.b. Fetch Unit Control Logic 20214 Operation
 - a.a.a. Prefetcher 20264, Instruction Buffer 20262, Parser 20264, Operation Code Register 20268, CPC 20270, IPC 20272, and EPC 20274 (Fig. 241)
 - 55 b.b.b. Fetch Unit Dispatch Table 11010, Execute Unit Dispatch Table 20266, and Operation Code Register 20268 (Fig. 242)
 - c.c.c. Next Address Generator 24310 (Fig. 243)
 - cc. FUCTL 20214 Circuitry for CS 10110 Internal Mechanisms (Figs. 244-250)
 - a.a.a. State Logic 20294 (Figs. 244A—244Z)
 - 60 b.b.b. Event Logic 20284 (Figs. 245, 246, 247, 248)
 - c.c.c. Fetch Unit S-Interpreter Table 11012 (Fig. 249)
 - d.d. CS 10110 Internal Mechanism Control
 - a.a.a. Return Control Word Stack 10358 (Fig. 251)
 - b.b.b. Machine Control Block (Fig. 252)
 - 65 c.c.c. Register Address Generator 20288 (Fig. 253)

EP 0 067 556 B1

- d.d.d. Timers 20296 (Fig. 254)
- e.e.e. Fetch Unit 10120 Interface to Execute Unit 10122
- C. Execute Unit 10122 (Figs. 203, 255-268)
 - a. General Structure of Execute Unit 10122
 - 1. Execute Unit I/O 20312
 - 2. Execute Unit Control Logic 20310
 - 3. Multiplier Logic 20314
 - 4. Exponent Logic 20316
 - 5. Multiplier Control 20318
 - 6. Test and Interface Logic 20320
 - b. Execute Unit 10122 Operation (Fig. 255)
 - 1. Execute Unit Control Logic 20310 (Fig. 255)
 - a.a. Command Queue 20342
 - b.b. Command Queue Event Control Store 25514 and Command Queue Event Address Control Store 25516
 - c.c. Execute Unit S-Interpreter Table 20344
 - d.d. Microcode Control Decode Register 20346
 - e.e. Next Address Generator 20340
 - 2. Operand Buffer 20322 (Fig. 256)
 - 3. Multiplier 20314 (Figs. 257, 258)
 - a.a. Multiplier 20314 I/O Data Paths and Memory (Fig. 257)
 - a.a.a. Container Size Check
 - b.b.b. Final Result Output Multiplexer 20324
 - 4. Test and Interface Logic 20320 (Figs. 260-268)
 - a.a. FU 10120/EU 10122 Interface
 - a.a.a. Loading of Command Queue 20342 (Fig. 260)
 - b.b.b. Loading of Operand Buffer 20320 (Fig. 261)
 - c.c.c. Storeback (Fig. 262)
 - d.d.d. Test Conditions (Fig. 263)
 - e.e.e. Exception Checking (Fig. 264)
 - f.f.f. Idle Routine
 - g.g.g. EU 10122 Stack Mechanisms (Figs. 265, 266, 267)
 - h.h.h. Loading of Execute Unit S-Interpreter Table 20344 (Fig. 268)
- D. I/O System 10116 (Figs. 204, 206, 269)
 - a. I/O System 10116 Structure (Fig. 204)
 - b. I/O System 10116 Operation (Fig. 269)
 - 1. Data Channel Devices
 - 2. I/O Control Processor 20412
 - 3. Data Mover 20410 (Fig. 269)
 - a.a. Input Data Buffer 20440 and Output Data Buffer 20442
 - b.b. Priority Resolution and Control 20444 (Fig. 269)
- E. Diagnostic Processor 10118 (Fig. 101, 205)
- F. CS 10110 Micromachine Structure and Operation (Figs. 270—274)
 - a. Introduction
 - b. Overview of Devices Comprising FU Micromachine (Fig. 270)
 - 1. Devices Used By Most Microcode
 - a.a. MOD Bus 10144, JPD Bus 10142, and DB Bus 27021
 - b.b. Microcode Addressing
 - c.c. Descriptor Processor 20218 (Fig. 271)
 - d.d. EU 10122 Interface
 - 2. Specialized Micromachine Devices
 - a.a. Instruction Stream Reader 27001
 - b.b. SOP Decoder 27003
 - c.c. Name Translation Unit 27015
 - d.d. Memory Reference Unit 27017
 - e.e. Protection Unit 27019
 - f.f. KOS Micromachine Devices
 - c. Micromachine Stacks and Microroutine Calls and Returns (Figs. 272, 273)
 - 1. Micromachine Stacks (Fig. 272)
 - 2. Micromachine Invocations and Returns
 - 3. Means of Invoking Microroutines
 - 4. Occurrence of Event Invocations (Fig. 273)
 - d. Virtual Micromachines and Monitor Micromachine
 - 1. Virtual Mode
 - 2. Monitor Micromachine

EP 0 067 556 B1

- e. Interrupt and Fault Handling
 - 1. General Principles
 - 2. Hardware Interrupt and Fault Handling in CS 10110
 - 3. Monitor Mode: Differential Masking and Hardware Interrupt Handling
- 5 g. FU Micromachine and CS 10110 Subsystems
- 3. Namespace, S-Interpreters and Pointers (Figs. 301—307, 274)
- A. Pointers and Pointer Resolution (Figs. 301, 302)
 - a. Pointer Formats (Fig. 301)
 - 10 b. Pointers in FU 10120 (Fig. 302)
 - c. Descriptor to Pointer Conversion
- B. Namespace and the S-Interpreters (Figs. 303—307)
 - a. Procedure Object 606 Overview (Fig. 303)
 - 15 b. Namespace
 - 1. Name Resolution and Evaluation
 - 2. The Name Table (Fig. 304)
 - 3. Architectural Base Pointers (Figs. 305, 306)
 - a.a. Resolving and Evaluating Names (Fig. 305)
 - 20 b.b. Implementation of Name Evaluation and Name Resolve in CS 10110
 - c.c. Name Cache 10226 Entries (Fig. 306)
 - d.d. Name Cache 10226 Hits
 - e.e. Name Cache 10226 Misses
 - f.f. Flushing Name Cache 10226
 - 25 g.g. Fetching the Instruction Stream
 - h.h. Parsing the Instruction Stream
 - c. The S-Interpreters (Fig. 307)
 - 1. Translating SIP into a Dialect Number (Fig. 307)
 - 2. Dispatching
- 30 4. The Kernel Operation System
- A. Introduction
 - a. Operating Systems (Fig. 401)
 - 1. Resources Controlled by Operating Systems (Fig. 402)
 - b. The Operating System in CS 10110
 - 35 c. Extended Operating System and the Kernel Operating System (Fig. 403)
- B. Objects and Object Management (Fig. 404)
 - a. Objects and User Programs (Fig. 405)
 - b. UIDs 40401 (Fig. 406)
 - c. Object Attributes
 - 40 d. Attributes and Access Control
 - e. Implementation of Objects
 - 1. Introduction (Figs 407, 408)
 - 2. Objects in Secondary Storage 10124 (Figs. 409, 410).
 - a.a. Representation of an Object's Contents on Secondary Storage 10124
 - 45 b.b. LAUD 40903 (Figs. 411, 412)
 - 3. Active Objects (Fig. 413)
 - a.a. UID 40401 to AON 41304 Translation
- C. The Access Control System
 - a. Subjects
 - 50 b. Domains
 - c. Access Control Lists
 - 1. Subject Templates (Fig. 416)
 - 2. Primitive Access Control Lists (PACLs)
 - 3. APAM 10918 and Protection Cache 10234 (Fig. 421)
 - 55 4. Protection Cache 10234 and Protection Checking (Fig. 422)
- D. Processes
 - 1. Synchronization of Processes 610 and Virtual Processors 612
 - a. Event Counters 44801, Await Entries 44804, and Await Tables (Fig. 448, 449)
 - b. Synchronization with Event Counters 44801 and Await Entries 44804
- 60 E. Virtual processors 612 (Fig. 453)
 - a. Virtual Processor Management (Fig. 453)
 - b. Virtual Processors 612 and Synchronization (Fig. 454)
- F. Process 610 Stack Manipulation
 - 1. Introduction to Call and Return
 - 65 2. Macrostacks (MAS) 502 (Fig. 467)

- a.a. MAS Base 10410 (Fig. 468)
- b.b. Per-domain Data Area 46853 (Fig. 468)
- c.c. MAS Frame 46709 Detail (Fig. 469)
- 3. SS 504 (Fig. 470)
 - a.a. SS Base 47001 (Fig. 471)
 - b.b. SS Frames 47003 (Fig. 471)
 - a.a.a. Ordinary SS Frame Headers 10514 (Fig. 471)
 - b.b.b. Detailed Structure of Macrostate 10516 (Fig. 471)
 - c.c.c. Cross-domain SS Frames 47039 (Fig. 471)
- 4. Portions of Procedure Object 608 Relevant to Call and Return (Fig. 472)
- 5. Execution of Mediated Calls
 - a.a. Mediated Call SInS
 - b.b. Simple Mediated Calls (Figs. 270, 468, 469, 470, 471, 472)
 - c.c. Invocations of Procedures 602 Requiring SEBs 46864 (Figs. 270, 468, 469, 470, 471, 472)
 - d.d. Cross-Procedure Object Calls (Figs. 270, 468, 469, 470, 471, 472)
 - e.e. Cross-Domain Calls (Figs. 270, 408, 418, 468, 469, 470, 471, 472)
 - f.f. Failed Cross-Domain Calls (Figs. 270, 468, 469, 470, 471, 472)
- 6. Neighborhood Calls (Figs. 468, 469, 472)

INTRODUCTORY OVERVIEW

The following overview will first briefly describe the overall physical structure and operation of a presently preferred embodiment of a digital computer system incorporating the present invention. Then certain operating features of that computer system will be individually described. Next, overall operation of the computer system will be described in terms of those individual features.

A. Hardware Overview (Fig. 1)

Referring to Fig. 1, a block diagram of Computer System (CS) 101 incorporating the present invention is shown. Major elements of CS 101 are I/O System (IOS) 116, Memory (MEM) 112, and Job Processor (JP) 114. JP 114 is comprised of a Fetch Unit (FU) 120 and an Execute Unit (EU) 122. CS 101 may also include a Diagnostic Processor (DP), not shown or described in the instant description.

Referring first to IOS 116, a primary function of IOS 116 is control of transfer of information between MEM 112 and the outside world. Information is transferred from MEM 112 to IOS 116 through IOM Bus 130, and from IOS 116 to MEM 112 through MIO Bus 129. IOMC Bus 131 is comprised of bi-directional control signals coordinating operation of MEM 112 and IOS 116. IOS 116 also has an interface to FU 120 through IOJP Bus 132. IOJP Bus 132 is a bi-directional control bus comprised essentially of two interrupt lines. These interrupt lines allow FU 120 to indicate to IOS 116 that a request for information by FU 120 has been placed in MEM 112, and allows IOS 116 to inform FU 120 that information requested by FU 120 has been transferred into a location in MEM 112. MEM 112 is CS 101's main memory and serves as the path for information transfer between the outside world and JP 114. MEM 112 provides instructions and data to FU 120 and EU 122 through Memory Output Data (MOD) Bus 140 and receives information from FU 120 and EU 122 through Job Processor Data (JPD) Bus 142. FU 120 submits read and write requests to MEM 112 through Physical Descriptor (PD) Bus 146.

JP 114 is CS 101's CPU and, as described above, is comprised of FU 120 and EU 122. A primary function of FU 120 is executing operations of user's programs. As part of this function, FU 120 controls transfer of instructions and data from MEM 112 and transfer of results of JP 114 operations back to MEM 112. FU 120 also performs operating system type functions, and is capable of operating as a complete, general purpose CPU. EU 122 is primarily an arithmetic and logic unit provided to relieve FU 120 of certain arithmetic operations. FU 120, however, is capable of performing EU 122 operations. In alternate embodiments of CS 101, EU 122 may be provided only as an option for users having particular arithmetic requirements. Coordination of FU 120 and EU 122 operations is accomplished through FU/EU (FUEU) Bus 148, which includes bi-directional control signals and mutual interrupt lines. As described further below, both FU 120 and EU 122 contain register file arrays referred to respectively as CRF and ERF, in addition to registers associated with, for example, ALUs.

A primary feature of CS 101 is that IOS 116, MEM 112, FU 120 and EU 122 each contain separate and independent microinstruction control, so that IOS 116, MEM 112, and EU 122 operate asynchronously under the general control of FU 120. EU 122, for example, may execute a complex arithmetic operation upon receipt of data and a single, initial command from FU 120.

Having briefly described the overall structure and operation of CS 101, certain features of CS 101 will be individually further described next below.

B. Individual Operating Features (Figs. 2, 3, 4, 5, 6)

1. Addressing (Fig. 2)

Referring to Fig. 2, a diagrammatic representation of portions of CS 101's addressing structure is shown. CS 101's addressing structure is based upon the concept of Objects. An Object may be regarded as a

container for holding a particular type of information. For example, one type of Object may contain data while another type of Object may contain instructions or procedures, such as a user program. Still another type of Object may contain microcode. In general, a particular Object may contain only one type or class of information. An Object may, for example, contain up to 232 bits of information, but the actual size of a particular Object is flexible. That is, the actual size of a particular Object will increase as information is written into that Object and will decrease as information is taken from that Object. In general, information in Objects is stored sequentially, that is without gaps.

Each Object which can ever exist in any CS 101 system is uniquely identified by a serial number referred to as a Unique Identifier (UID). A UID is a 128 bit value comprised of a serial number dependent upon, for example, the particular CS 101 system and user, and a time code indicating time of creation of that Object. UIDs are permanently assigned to Objects, no two Objects may have the same UID, and UIDs may not be reused. UIDs provide an addressing base common to all CS 101 systems which may ever exist, through which any Object ever created may be permanently and uniquely identified.

As described above, UIDs are 128 bit values and are thus larger than may be conveniently handled in present embodiments of CS 101. In each CS 101, therefore, those Objects which are active (currently being used) in that system are assigned 14 bit Active Object Numbers (AONs). Each Object active in that system will have a unique AON. Unlike UIDs, AONs are only temporarily assigned to particular Objects. AONs are valid only within a particular CS 101 and are not unique between systems. An Object need not physically reside in a system to be assigned an AON, but can be active in that system only if it has been assigned an AON.

A particular bit within a particular Object may be identified by means of a UID address or an AON address. In CS 101, AONs and AON addresses are valid only within JP 114 while UIDs and UID addresses are used in MEM 112 and elsewhere. UID and AON addresses are formed by appending a 32 bit Offset (O) field to that Object's UID or AON. O fields indicate offset, or location, of a particular bit relative to the start of a particular Object.

Segments of information (sequences of information bits) within particular Objects may be identified by means of descriptors. A UID descriptor is formed by appending a 32 bit Length (L) field of a UID address. An AON, or logical descriptor is formed by appending a 32 bit L field to an AON address. L fields identify length of a segment of information bits within an Object, starting from the information bit identified by the UID or AON address. In addition to length information, UID and logical descriptors also contain Type fields containing information regarding certain characteristics of the information in the information segment. Again, AON based descriptors are used within JP 114, while UID based descriptors are used in MEM 112.

Referring to Figs. 1 and 2 together, translation between UID addresses and descriptors and AON addresses and descriptors is performed at the interface between MEM 112 and JP 114. That is, addresses and descriptors within JP 114 are in AON form while addresses and descriptors in MEM 112, IOS 116, and the external world are in UID form. In other embodiments of CS 101 using AONs, transformation from UID to AON addressing may occur at other interfaces, for example at the IOS 116 to MEM 112 interface, or at the IOS 116 to external world interface. Other embodiments of CS 101 may use UIDs throughout, that is not use AONs even in JP 114.

Finally, information within MEM 112 is located through MEM 112 Physical Addresses identifying particular physical locations within MEM 112's memory space. Both IOS 116 and JP 114 address information within MEM 112 by providing physical addresses to MEM 112. In the case of physical addresses provided by JP 114, these addresses are referred to as Physical Descriptors (PDs). As described below, JP 114 contains circuitry to translate logical descriptors into physical descriptors.

2. S-Language Instructions and Namespace Addressing (Fig. 3)

CS 101 is both an S-Language machine and a Namespace machine. That is, operations to be executed by CS 101 are expressed as S-Language Operations (SOPs) while operands are identified by Names. SOPs are of a lower, more detailed, level than user language instructions, for example FORTRAN and COBOL, but of a higher level than conventional machine language instructions. SOPs are specific to particular user languages rather than a particular embodiment of CS 101, while conventional machine language instructions are specific to particular machines. SOPs are in turn interpreted and executed by microcode. There will be an S-Language Dialect, a set of SOPs, for each user languages. CS 101, for example, may have SOP Dialects for COBOL, FORTRAN, and SPL. A particular distinction of CS 101 is that all SOPs are of a uniform, fixed length, for example 16 bits. CS 101 may generally contain one or more sets of microcode for each S-Language Dialect. These microcode Dialect Sets may be completely distinct, or may overlap where more than one SOP utilizes the same microcode.

As stated above, in CS 101 all operands are identified by Names, which are 8, 12, or 16 bit numbers. CS 101 includes one or more "Name Tables" containing an Entry for each operand Name appearing in programs currently being executed. Each Name Table Entry contains information describing the operand referred to by a particular Name, and the directions necessary for CS 101 to translate that information into a corresponding logical descriptor. As previously described, logical descriptors may then be transformed into physical descriptors to read and write operands from or to MEM 112. As described above, UIDs are unique for all CS 101 systems and AONs are unique within individual CS 101 systems. Names, however, are unique only within the context of a user's program. That is, a particular Name may appear in two different

EP 0 067 556 B1

user's programs and, within each program, will have different Name Table Entries and will refer to different operands.

CS 101 may thereby be considered as utilizing two sets of instructions. A first set is comprised of SOPs, that is instructions selecting algorithms to be executed. The second set of instructions are comprised of Names, which may be regarded as entry points into tables of instructions for making references regarding operands.

Referring to Fig. 3, a diagrammatic representation of CS 101 instruction stream is shown. A typical SIN is comprised of an SOP and may include one or more Names referring to operands. SOPs and Names allow user's programs to be expressed in very compact code. Fewer SOPs than machine language instructions are required to express a user's program. Also, use of SOPs allows easier and simpler construction of compilers, and facilitates adaption of CS 101 systems to new user languages. In addition, use of Names to refer to operands means that SOPs are independent of the form of the operands upon which they operate. This in turn allows for more compact code in expressing user programs in that SOPs specifying operations dependent upon operand form are not required.

3. Architectural Base Pointer Addressing

As will be described further below, a user's program residing in CS 101 will include one or more Objects. First, a Procedure Object contains at least the SINs of the user's programs and a Name Table containing entries for operand Names of the program. The SINs may include references, or calls, to other Procedure Objects containing, for example, procedures available in common to many users. Second, a Static Data Area may contain static data, that is data having an existence for at least a single execution of the program. And third, a Macro-stack, described below, may contain local data, that is data generated during execution of a program. Each Procedure Object, the Static Data Area and the Macro-stack are individual Objects identified by UIDs and AONs and addressable through UID and AON addresses and descriptors.

Locations of information within a user's Procedure Objects, Static Data Area, and Macro-stack are expressed as offsets from one of three values, or base addresses, referred to as Architectural Base Pointers (ABPs). For example, location information in Name Tables is expressed as offsets from one of the ABPs. ABPs may be expressed as previously described.

The three ABPs are the Frame Pointer (FP), the Procedure Base Pointer (PBP), and the Static Data Pointer (SDP). Locations of data local to a procedure, for example in the procedure's Macrostack, are described as offsets from FP. Locations of non-local data, that is Static Data, are described as offsets from SDP. Locations of SINs in Procedure Objects are expressed as offsets from PBP; these offsets are determined as a Program Counter (PC) value. Values of the ABPs vary during program execution and are therefore not provided by the compiler converting a user's high level language program into a program to be executed in a CS 101 system. When the program is executed, CS 101 provides the proper values for the ABPs. When a program is actually being executed, the ABP's values are stored in FU 120's GRF.

Other pointers are used, for example, to identify the top frame of CS 101's Secure Stack (a microcode level stack described below) or to identify the microcode Dialect currently being used in execute the SINs of a procedure. These pointers are similar to FP, SDP, and PBP.

4. Stack Mechanisms (Fig. 4—5)

Referring to Fig. 4 and 4A, diagrammatic representations of various control levels and stack mechanisms of, respectively, conventional machines and CS 101, are shown. Referring first to Fig. 4, top level of control is provided by User Language Instructions 402, for example in FORTRAN or COBOL. User Language Instructions 402 are converted into a greater number of more detailed Machine Language Instructions 404, used within a machine to execute user's programs. Within the machine, Machine Language Instructions 404 are interpreted and executed by Microcode Instructions 406, that is sequences of microinstructions which in turn directly control Machine Hardware 408. Some conventional machines may include a Stack Mechanism 410 used to save current machine state, that is current microinstruction and contents of various machine registers, if a current Machine Language Instruction 404 cannot be executed or is interrupted. In general, machine state on the microcode and hardware level is not saved. Execution of a current Machine Language Instruction 404 is later resumed at start of the microinstruction sequence for executing that Machine Language Instruction 404.

Referring to Fig. 4A, top level control in CS 101 is by User Language Instructions 412 as in a conventional machine. In CS 101, however, User Language Instructions 412 are translated into SOPs 414 which are of a higher level than conventional machine language instructions. In general, a single User Language Instruction 412 is transformed into at most two or three SOPs 414, as opposed to an entire sequence of conventional Machine Language Instructions 404. SOPs 414 are interpreted and executed by Microcode Instructions 416 (sequences of microinstructions) which directly control CS 101 Hardware 418. CS 101 includes a Macro-stack Mechanism (MAS) 420, at SOPs 414 level, which is comparable to but different in construction and operation from a conventional Machine Language Stack Mechanism 410. CS 101 also includes Micro-code Stack Mechanisms 422 operating at Microcode 416 level, so that execution of an interrupted microinstruction of a microinstruction sequence may be later resumed with the particular microinstruction which was active at the time of the interrupt. CS 101 is therefore more efficient in handling

EP 0 067 556 B1

interrupts in that execution of microinstruction sequences is resumed from the particular point that a microinstruction sequence was interrupted, rather from the beginning of that sequence. As will be described further below, CS 101's Micro-code Stack Mechanisms 422 on microcode level is effectively comprised of two stack mechanisms. The first stack is Micro-Instruction Stack (MIS) 424 while the second stack is referred to as Monitor Stack (MOS) 426. CS 101 SIN Microcode 428 and MIS 424 are primarily concerned with execution of SOPs of user's programs. Monitor Microcode 430 and MOS 426 are concerned with operation of certain CS 101 internal functions.

Division of CS 101's microcode stacks into an MIS 424 and a MOS 426 illustrates a further feature of CS 101. In conventional machines, monitor functions may be performed by a separate CPU operating in conjunction with the machine's primary CPU. In CS 101, a single hardware CPU is used to perform both functions with actual execution of both functions performed by separate groups of microcode. Monitor microcode operations may be initiated either by certain SInS 414 or by control signals generated directly by CS 101's Hardware 418. Invocation of Monitor Microcode 430 by Hardware 418 generated signals insures that CS 101's monitor functions may always be invoked.

Referring to Fig. 5, a diagramic representation of CS 101's stack mechanisms for a *single user's program, or procedure*, is shown. Basically, and with exception of MOS 426, CS 101's stacks reside in MEM 112 with certain portions of those stacks accelerated into FU 120 and EU 122 to enhance speed of operation.

Certain areas of MEM 112 storage space are set aside to contain Macro-Stacks (MASs) 502, stack mechanisms operating on the SInS level, as described above. Other areas of MEM 112 are set aside to contain Secure Stack (SS) 504, operating on the microcode level, as described above and of which MIS 424 is a part.

As described further below, both FU 120 and EU 122 contain register file arrays, referred to respectively as GRF and ERF, in addition to registers associated with, for example, ALUs. Referring to FU 120, shown therein is FU 120's GRF 506. GRF 506 is horizontally divided into three areas. A first area, referred to as General Registers (GRs) 508 may in general be used in the same manner as registers in a conventional machine. A second area of GRF 506 is Micro-Stack (MIS) 424, and is set aside to contain a portion of a Process's SS 504. A third portion of GRF 506 is set aside to contain MOS 426. Also indicated in FU 120 is a block referred to as Microcode Control State (mCS) 510. mCS 510 represents registers and other FU 120 hardware containing current operating state of FU 120 on the microinstruction and hardware level. mCS 510 may include, for example, the current microinstruction controlling operation of FU 120.

Referring to EU 122, indicated therein is a first block referred to as Execute Unit State (EUS) 512 and a second block referred to as SOP Stack 514. EUS 512 is similar to mCS 510 in FU 120 and includes all registers and other EU 122 hardware containing information reflecting EU 122's current operating state. SOP Stack 518 is a portion of EU 122's ERF 516 which has been set aside as a stack mechanism to contain a portion of a process's SS 504 pertaining to EU 122 operations.

Considering first MASs 502, as stated above MASs 502 operate generally upon the SInS level. MASs 502 are used in general to store current state of a process's (defined below) execution of a user's program.

Referring next to MIS 424, in a present embodiment of CS 101 that portion of GRF 506 set aside to contain MIS 424 may have a capacity of eight stack frames. That is, up to 8 microinstruction level interrupts or calls pertaining to execution of a user's program may be stacked within MIS 424. Information stored in MIS 424 stack frames is generally information from GR 508 and MCS 510. MIS 424 stack frames are transferred between MIS 424 and SS 504 such that at least one frame, and no more than 8 frames, of SS 504 reside in GRF 506. This insures that at least the top-most frames of a process's SS 504 are present in FU 120, thereby enhancing speed of operation of FU 120 by providing rapid access to those top frames. SS 504, residing in MEM 112, may contain, for all practical purposes, an unlimited number of frames so that MIS 424 and SS 504 appear to a user to be effectively an infinitely deep stack.

MOS 426 resides entirely in FU 120 and, in a present embodiment of CS 101, may have a capacity of 8 stack frames. A feature of CS 101 operation is that CS 101 mechanisms for handling certain events or interrupts should not rely in its operation upon those portions of CS 101 whose operation has resulted in those faults or interrupts. Among events handled by CS 101 monitor microcode, for example, are MEM 112 page faults. An MEM 112 page fault occurs whenever FU 120 makes a reference to data in MEM 112 and that data is not in MEM 112. Due to this and similar operations, MOS 426 resides entirely in FU 120 and thus does not rely upon information in MEM 112.

As described above, GRs 508, MIS 424, and MOS 426 each reside in certain assigned portions of GRF 506. This allows flexibility in modifying the capacity of GRs 508, MIS 424, and MOS 426 as indicated by experience, or to modify an individual CS 101 for particular purposes.

Referring finally to EU 122, EUS 512 is functionally a part of a process's SS 504. Also as previously described, EU 122 performs arithmetic operations in response to SInS and may be interrupted by FU 120 to aid certain FU 120 operations. EUS 512 allows stacking of interrupts. For example, FU 120 may first interrupt an arithmetic SOP to request EU 122 to aid in evaluation of a Name Table Entry. Before that first interrupt is completed, FU 120 may interrupt again, and so on.

SOP Stack 514, is a single frame stack for storing current state of EU 122 when an interrupt interrupts execution of an arithmetic SOP. An interrupted SOP's state is transferred into SOP Stack 514 and the interrupt begins execution in EUS 512. Upon occurrence of a second interrupt (before the first interrupt is completed) EU's first interrupt state is transferred from EUS 512 to a stack frame in SS 504, and execution

of the second interrupt begins in EUS 512. If a third interrupt occurs before completion of second interrupt, EU's second interrupt state is transferred from EUS 512 to another stack frame in SS 504 and execution of the third interrupt is begun in EUS 512; and so on. EUS 512 and SS 504 thus provide an apparently infinitely deep microstack for EU 122. Assuming that the third interrupt is completed, state of second interrupt is transferred from SS 504 to EUS 512 and execution of second interrupt resumed. Upon completion of second interrupt, state of first interrupt is transferred from SS 504 to EUS 512 and completed. After completion of first interrupt, state of the original SOP is transferred from SOP Stack 514 to EUS 512 and execution of that SOP resumed.

10 C. Procedure Processes, and Virtual Processors (Fig. 6)

Referring to Fig. 6, a diagrammatic representation of procedures, processes, and virtual processes is shown. As described above, a user's program to be executed is compiled to result in a Procedure 602. A Procedure 602 includes a User's Procedure Object 604 containing the SOPs of the user's program and a Name Table containing Entries for operand Names of the user's program, and a Static Data Area 606. A Procedure 602 may also include other Procedure Objects 608, for example utility programs available in common to many users. In effect, a Procedure 602 contains the instructions (procedures) and data of a user's program.

A Process 610 includes, as described above, a Macro Stack (MAS) 502 storing state of execution of a user's Procedure 602 at the SOP level, and a Secure Stack (SS) 504 storing state of execution of a user's Procedure 602 at the microcode level. A Process 610 is associated with a user's Procedure 602 through the ABPs described above and which are stored in the MAS 502 of the Process 610. Similarly, the MAS 502 and SS 504 of a Process 610 are associated through non-architectural pointers, described above. A Process 602 is effectively a body of information linking the resources, hardware, microcode, and software, of CS 101 to a user's Procedure 602. In effect, a Process 610 makes the resources of CS 101 available to a user's Procedure 602 for executing of that Procedure 602. CS 101 is a multi-program machine capable of accommodating up to, for example, 128 processes 610 concurrently. The number of Processes 610 which may be executed concurrently is determined by the number of Virtual Processors 612 of CS 101. There may be, for example, up to 16 Virtual Processors 612.

As indicated in Fig. 6, a Virtual Processor 612 is comprised of a Virtual Processor State Block (VPSB) 614 associated with the SS 504 of a Process 612. A VPSB 614 is, in effect, a body of information accessible to CS 101's operating system and through which CS 101's operating system is informed of, and provided with access to, a Process 610 through that process 610's SS 504. A VPSB 614 is associated with a particular Process 610 by writing information regarding that Process 610 into that VPSB 614. CS 101's operating system may, by gaining access to a Process 610 through an associated PSB 614, read information, such as ABP's, from that Process 610 to FU 120, thereby swapping that Process 610 onto FU 120 for execution. It is said that a Virtual Processor 612 thereby executes a process 610; a Virtual Processor 612 may be regarded therefor, as a processor having "Virtual", or potential, existence which becomes "real" when its associated Process 610 is swapped into FU 120. In CS 101, as indicated in Fig. 6, only one Virtual Processor 612 may execute on FU 120 at a time and the operating system selects which Virtual Processor 612 will execute on FU 120 at any given time. In addition, CS 101's operating system selects which Processes 610 will be associated with the available Virtual Processors 612.

Having briefly described certain individual structural and operating features of CS 101, the overall operation of CS 101 will be described in further detail next below in terms of these individual features.

45 D. CS 101 Overall Structure and Operation (Figs. 7, 8, 9, 10, 11, 12, 13, 14, 15)

1. Introduction (Fig. 7)

As indicated in Fig. 7, CS 101 is a multiple level system wherein operations in one level are generally transparent to higher levels. User 701 does not see the S-Language, addressing, and protection mechanisms defined at Architectural Level 708. Instead, he sees User Interface 709, which is defined by Compilers 702, Binder 703, and Extended (high level) Operating System (EOS) 704. Compilers 702 translate high-level language code into SInS and Binder 703 translates symbolic Names in programs into UID-offset addresses.

As Fig. 7 shows, Architectural Level 708 is not defined by FU 120 Interface 711. Instead, the architectural resources level are created by S-Language Interpreted SInS when a program is executed; Name Interpreter 715 operates under control of S-Language Interpreters 705 and translates Names into logical descriptors. In CS 101, both S-Language Interpreters 705 and Name Interpreter 715 are implemented as microcode which executes on FU 120. S-Language Interpreters 705 may also use EU 122 to perform calculations. A Kernel Operating System (KOS) provides CS 101 with UID-offset addressing, objects, access checking, processes, and virtual processors, described further below. KOS has three kinds of components: KOS Microcode 710, KOS Software 706, and KOS Tables in MEM 112. KOS 710 components are microcode routines which assist FU 120 in performing certain required operations. Like other high-level language routines, KOS 706 components contain SInS which are interpreted by S-Interpreter Microcode 705. Many KOS High-Level Language Routines 706 are executed by special KOS processes; others may be executed by any process. Both KOS High-Level Language Routines 706 and KOS Microcode 710 manipulate KOS Tables in MEM 112.

EP 0 067 556 B1

FU 120 Interface 711 is visible only to KOS and to S-Interpreter Microcode 705. For the purposes of this discussion, FU 120 may be seen as a processor which contains the following main elements:

A Control Mechanism 725 which executes microcode stored in Writable Control Store 713 and manipulates FU 120 devices as directed by this microcode.

A GRF 506 containing registers in which data may be stored.

A Processing Unit 715.

All microcode which executes on FU 120 uses these devices; there is in addition a group of devices for performing special functions; these devices are used only by microcode connected with those functions. The microcode, the specialized devices, and sometimes tables in MEM 112 make up logical machines for performing certain functions. These machines will be described in detail below.

In the following, each of the levels illustrated in Fig. 7 will be discussed in turn. First, the components at User Interface 709 will be examined to see how they translate user programs and requests into forms usable by CS 101. Then the components below the User Interface 709 will be examined to see how they create logical machines for performing CS 101 operations.

2. Compilers 702 (Fig. 7)

Compilers 702 translate files containing the highlevel language code written by User 701 into Procedure Objects 608. Two components of a Procedure Object 608 are code (SOPs) and Names, previously described. SOPs represent operations, and the Names represent data. A single SIN thus specifies an operation to be performed on the data represented by the Names.

3. Binder 703 (Fig. 7)

In some cases, Compiler 702 cannot define locations as offsets from an ABP. For example, if a procedure calls a procedure contained in another procedure object, the location to which the call transfers control cannot be defined as an offset from the PBP used by the calling procedure. In these cases, the compiler uses symbolic Names to define the locations. Binder 703 is a utility which translates symbolic Names into UID-offset addresses. It does so in two ways: by combining separate Procedure Objects 608 into a single large Procedure Object 608, and then redefining symbolic Names as offsets from that Procedure Object 608's ABPs, or by translating symbolic Names when the program is executed. In the second case, Binder 703 requires assistance from EOS 704.

4. EOS 704 (Fig. 7)

EOS 704 manages the resources that User 701 requires to execute his programs. From User 701's point of view, the most important of these resources are files and processes. EOS 704 creates files by requesting KOS to create an object and then mapping the file onto the object. When a User 701 performs an operation on a file, EOS 704 translates the file operation into an operation on an object. KOS creates them at EOS 704's request and makes them available to EOS 704, which in turn makes them available to User 701. EOS 704 causes a process to execute by associating it a Virtual Processor 612. In logical terms, a Virtual Processor 612 is the means which KOS provides EOS 704 for executing processes 610. As many Processes 610 may apparently execute simultaneously in CS 101 as there are Virtual Processors 612. The illusion of simultaneous execution is created by multiplexing JP 114 among the Virtual processors; the manner in which Processes 610 and Virtual Processors 610 are implemented will be explained in detail below.

5. KOS and Architectural Interface 708 (Fig. 7)

S-Interpreter Microcode 710 and Name Interpreter Microcode 715 require an environment provided by KOS Microcode 710 and KOS Software 706 to execute SINS. For example, as previously explained, Names and program locations are defined in terms of ABPs whose values vary during execution of the program. The KOS environment provides values for the ABPs, and therefore makes it possible to interpret Names and program locations as locations in MEM 112. Similarly, KOS help is required to transform logical descriptors into references to MEM 112 and to perform protection checks.

The environment provided by KOS has the following elements:

A Process 610 which contains the state of an execution of the program for a given User 701.

A Virtual Processor 612 which gives the Process 610 access to JP 114.

An Object Management System which translates UIDs into values that are usable inside JP 114.

A Protection System which checks whether a Process 610 has the right to perform an operation on an Object.

A Virtual Memory Management System which moves those portions of Objects which a Process 610 actually references from the outside world into MEM 112 and translates logical descriptors into physical descriptors.

In the following, the logical properties of this environment and the manner in which a program is executed in it will be explained.

6. Processes 610 and Virtual Processors 612 (Fig. 8)

Processes 610 and Virtual Processors 612 have already been described in logical terms; Fig. 8 gives a high-level view of their physical implementation.

Fig. 8 illustrates the relationship between Processes 610, Virtual Processors 612, and JP 114. In physical terms, a Process 610 is an area of MEM 112 which contains the current state of a user's execution of a program. One example of such state is the current values of the ABPs and a program Counter (PC). Given the current value of the PBP and the PC, the next SOP in the program can be executed; similarly, given the current values of SDP and FP, the program's Names can be correctly resolved. Since the Process 610 contains the current state of a program's execution, the program's physical execution can be stopped and resumed at any point. It is thus possible to control program execution by means of the Process 610.

As already mentioned, a process 610's execution proceeds only when KOS has bound it to a Virtual Processor 612, that is, an area of MEM 112 containing the state required to execute microinstructions on JP 114 hardware. The operation of binding is simply a transfer of Process 610 state from the Process 610's area of MEM 112 to a Virtual Processor 612's area of MEM 112. Since binding and unbinding may take place at any time, EOS 704 may multiplex Processes 610 among Virtual Processors 612. In Fig. 8, there are more Processes 610 than there are Virtual Processors 612. The physical execution of a Process 610 on JP 114 takes place only while the Process 610's Virtual Processor 612 is bound to JP 114, i.e., when state is transferred from Virtual Processor 612's area of MEM 112 to JP 114's registers. Just as EOS 704 multiplexes Virtual Processors 612 among Processes 610, KOS multiplexes JP 114 among Virtual Processors 612. In Fig. 8, only one Process 610 is being physically executed. The means by which JP 114 is multiplexed among Virtual Processors 612 will be described in further detail below.

7. Processes 610 and Stacks (Fig. 9)

In CS 101 systems, a Process 610 is made up of six Objects: one Process Object 901 and Five Stack Objects 902 to 906. Fig. 9 illustrates a Process 610. Process Object 901 contains the information which EOS 704 requires to manage the Process 610. EOS 704 has no direct access to Process Object 901, but instead obtains the information it needs by means of functions provided to it by KOS 706, 710. Included in the information are the UIDs of Stack Objects 902 through 906. Stack Objects 902 to 906 contain the Process 610's state.

Stack Objects 902 through 905, are required by CS 101's domain protection method and comprise Process 610's MAS 502. Briefly, a domain is determined in part by operations performed when a system is operating in that domain. For example, the system is in EOS 704 domain when executing EOS 704 operations and in KOS 706, 710 domain when executing KOS 706, 710 operations. A Process 610 must have one stack for each domain it enters. In the present embodiment, the number of domains is fixed at four, but alternate embodiments may allow any number of domains, and correspondingly, any number of Stack Objects. Stack Object 906 comprises Process 610's Secure Stack 504 and is required to store state which may be manipulated only by KOS 706, 710.

Each invocation made by a Process 610 results in the addition of frames to Secure Stack 504 and to Macro-Stack 502. The state stored in the Secure Stack 504 frame includes the macrostate for the invocation, the state required to bind Process 610 to a Virtual Processor 612. The frame added to Macro-Stack 502 is placed in one of Stack Objects 902 through 905. Which Stack Objects 902 to 905 gets the frame is determined by the invoked procedure's domain of execution.

Fig. 9 shows the condition of a Process 610's MAS 502 and Secure Stack 504 after the Process 610 has executed four invocations. Secure Stack 504 has one frame for each invocation; the frames of Process 610's MAS 502 are found in Stack Objects 902, 904, and 905. As revealed by their locations, Frame 1 is for an invocation of a routine with KOS 706, 710 domain of execution, Frame 2 for an invocation of a routine with the EOS 704 domain of execution, and Frames 3 and 4 for invocations of routines with the User domain of execution. Process 610 has not yet invoked a routine with the Data Base Management System (DBMS) domain of execution. The frames in Stack Objects 902 through 905 are linked together, and a frame is added to or removed from Secure Stack 504 every time a frame is added to Stack Objects 902 through 905. MAS 502 and Secure Stack 504 thereby function as a single logical stack even though logically contained in five separate Objects.

B. Processes 610 and Calls (Figs. 10, 11)

In the CS 101, calls and returns are executed by KOS 706, 710. When KOS 706, 710 performs a call for a process, it does the following:

It saves the calling invocation's macrostate in the top frame of Secure Stack 504 (Fig. 9).

It locates the procedure whose Name is contained in the call. The location of the first SIN in the procedure becomes the new PBP.

Using information contained in the called procedure, KOS 706, 710 creates a new MAS 502 frame in the proper Stack Object 902 through 905 and a new Secure Stack 504 frame in Secure Stack 504. FP is updated to point to the new MAS 502. If necessary, SDP is also updated.

Once the values of the ABPs have been updated, the PC is defined, Names can be resolved, and execution of the invoked routine can commence. On a return from the invocation to the invoking routine, the stack frames are deleted and the ABPs are set to the values saved in the invoking routine's macrostate. The invoking routine then continues execution at the point following the invocation.

A Process 610 may be illustrated in detail by putting the FORTRAN statement A + B into a FORTRAN routine called EXAMPLE and invoking it from another FORTRAN routine named CALLER. To simplify the

EP 0 067 556 B1

example, it is assumed that CALLER and EXAMPLE both have the same domain of execution. The parts of EXAMPLE which are of interest look like this:

```
5      SUBROUTINE EXAMPLE (C)
      INTEGER X,C
      INTEGER A,B
10     ...
      A = B
      ...
15     RETURN
      END
```

The new elements are a formal argument, C, and a new local variable, X. A formal argument is a data item which receives its value from a data item used in the invoking routine. The formal argument's value thus varies from invocation to invocation. The portions of INVOKER which are of interest look like this:

```
      SUBROUTINE INVOKER
25     INTEGER Z
      ...
30     CALL EXAMPLE (Z)
      ...
      END
```

35 The CALL statement in INVOKER specifies the Name of the subroutine being invoked and the actual arguments for the subroutine's formal arguments. During the invocation, the subroutine's formal arguments take on the values of the actual arguments. Thus, during the invocation specified by this CALL statement, the formal argument C will have the value represented by the variable Z in INVOKER.

40 When INVOKER is compiled, the compiler produces a CALL SIN corresponding to the CALL statement. The CALL SIN contains a Name representing a pointer to the beginning of the called routine's location in a procedure object and a list of Names representing the call's actual arguments. When CALL is executed, the Names are interpreted to resolve the SIN's Names as previously described, and KOS 710 microcode to perform MAS 502 and Secure Stack 504 operations.

45 Fig. 10 illustrates the manner in which the KOS 710 call microcode manipulates MAS 502 and Secure Stack 504.

Fig. 10 includes the following elements:

Call Microcode 1001, contained in FU 120 Writable Control Store 1014.

PC Device 1002, which contains part of macrostate belonging to the invocation of INVOKER which is executing the CALL statement.

50 Registers in FU Registers 1014. Registers 1004 contents include the remainder of macrostate and the descriptors corresponding to Names for EXAMPLE's location and the actual argument Z.

Procedure Object 1006 contains the entries for INVOKER and EXAMPLE, their Name Tables, and their code.

55 Macro-Stack Object 1008 (MAS 502) and Secure Stack Object 1010 (Secure Stack 504) contain the stack frames for the invocations of INVOKER and EXAMPLE being discussed here. EXAMPLE's frame is in the same Macro-Stack object as INVOKER's frame because both routines are contained in the same Procedure Object 1006, and therefore have the same domain of execution.

60 KOS Call Microcode 1001 first saves the macrostate of INVOKER's invocation on Secure Stack 504. As will be discussed later, when the state is saved, KOS 706 Call Microcode 1001 uses other KOS 706 microcode to translate the location information contained in the macrostate into the kind of pointers used in MEM 112. Then Microcode 1001 uses the descriptor for the routine Name to locate the pointer to EXAMPLE's entry in Procedure Object 1006. From the entry, it locates pointers to EXAMPLE's Name Table and the beginning of EXAMPLE's code. Microcode 1001 takes these pointers, uses other KOS 706 microcode to translate them into descriptors, and places the descriptors in the locations in Registers 1004 reserved for the values of the PBP and NTP. It then updates the values contained in PC Device 1002 so that

when the call is finished, the next SIN to be executed will be the first SIN in EXAMPLE.

CALL Microcode 1001 next constructs the frames for EXAMPLE on Secure Stack 504 and Macro-Stack 502. This discussion concerns itself only with Frame 1102 on Macro-Stack 502. Fig. 11 illustrates EXAMPLE's Frame 1102. The size of Frame 1102 is determined by EXAMPLE's local variables (X, A, and B) and formal arguments (C). At the bottom of Frame 1102 is Header 1104. Header 1104 contains information used by KOS 706, 710 to manage the stack. Next comes Pointer 1106 to the location which contains the value represented by the argument C. In the invocation, the actual for C is the local variable Z in INVOKER. As is the case with all local variables, the storage represented by Z is contained in the stack frame belonging to INVOKER's invocation. When a name interpreter resolved C's name, it placed the descriptor in a register. Call Microcode 1001 takes this descriptor, converts it to a pointer, and stores the pointer above Header 1104.

Since the FP ABP points to the location following the last pointer to an actual argument, Call Microcode 1001 can now calculate that location, convert it into a descriptor, and place it in a FU Register 1004 reserved for FP. The next step is providing storage for EXAMPLE's local variables. EXAMPLE's procedure Object 1006 contains the size of the storage required for the local variables, so Call Microcode 1001 obtains this information from procedure Object 1006 and adds that much storage to Frame 1102. Using the new value of FP and the information contained in the Name Table Entries for the local data, Name Interpreter 715 can now construct descriptors for the local data. For example, A's entry in Name Table specified that it was offset 32 bits from FP, and was 32 bits long. Thus, its storage falls between the storage for X and B in Figure 11.

9. Memory References and the Virtual Memory Management System (Fig. 12, 13)

As already explained, a logical descriptor contains an AON field, an offset field, and a length field. Fig. 12 illustrates a Physical Descriptor. Physical Descriptor 1202 contains a Frame Number (FN) field, a Displacement (D) field, and a Length (L) field. Together, the Frame Number field and the Displacement field specify the location in MEM 112 containing the data, and the Length field specifies the length of the data.

As is clear from the above, the virtual memory management system must translate the AON-offset location contained in a logical descriptor 1204 into a Frame Number-Displacement location. It does so by associating logical pages with MEM 112 frames. (N.B: MEM 112 frames are not to be confused with stack frames). Fig. 13, illustrates how Macrostack 502 Object 1302 is divided into Logical Pages 1304 in secondary memory and how Logical Pages 1304 are moved onto Frames 1306 in MEM 112. A Frame 1306 is a fixed-size, contiguous area of MEM 112. When the virtual memory management system brings data into MEM 112, it does so in frame-sized chunks called Logical Pages 1308. Thus, from the virtual memory system's point of view, each object is divided into Logical Pages 1308 and the address of data on a page consists of the AON of the data's Object, the number of pages in the object, and its displacement on the page. In Fig. 13, the location of the local variable B of EXAMPLE is shown as it is defined by the virtual memory system. B's location is a UID and an offset, or, inside JP 114, an AON and an offset. As defined by the virtual memory system, B's location is the AON, the page number 1308, and a displacement within the page. When a process references the variable B, the virtual memory management system moves all of Logical Page 1308 into a MEM 112 Frame 1306. B's displacement remains the same, and the virtual memory system translates its Logical Page Number 1308 into the number of Frame 1306 in MEM 112 which contains the page.

The virtual memory management system must therefore perform two kinds of translations: (1) AON-offset addresses into AON-page number-displacement addresses, and (2) AON-page number into a frame number.

10. Access Control (Fig. 14)

Each time a reference is made to an Object, KOS 706, 710 checks whether the reference is legal. The following discussion will first present the logical structure of access control in CS 101, and then discuss the microcode and devices which implement it.

CS 101 defines access in terms of subjects, modes of access, and Object size. A process may reference a data item located in an Object if three conditions hold:

- 1) If the process's subject has access to the Object.
- 2) If the modes of access specified for the subject include those required to perform the intended operation.

- 3) If the data item is completely contained in the Object, i.e., if the data item's length added to the data item's offset do not exceed the number of bits in the Object.

The subjects which have access to an Object and the kinds of access they have to the Object are specified by a data structure associated with the Object called the Access Control List (ACL). An Object's size is one of its attributes. Neither an Object's size nor its ACL is contained in the Object. Both are contained in system tables, and are accessible by means of the Object's UID.

Fig. 14 shows the logical structure of access control in CS 101. Subject 1408 has four components: Principal 1404, Process 1405, Domain 1406, and Tag 1407. Tag 1407 is not implemented in a present embodiment of CS 101, so the following description will deal only with principal 1404, Process 1405, and Domain 1406.

EP 0 067 556 B1

Principal 1404 specifies a user for which the process which is making the reference was created; Process 1405 specifies the process which is making the reference; and, Domain 1406 specifies the domain of execution of the procedure which the process is executing when it makes the reference.

5 Each component of the Subject 1408 is represented by a UID. If the UID is a null UID, that component of the subject does not affect access checking. Non-null UIDs are the UIDs of Objects that contain information about the subject components. Principal Object 1404 contains identification and accounting information regarding system users, Process Object 1405 contains process management information, and Domain Object 1406 contains information about per-domain error handlers.

10 There may be three modes of accessing an Object 1410: read, write, and execute. Read and write are self-explanatory; execute is access which allows a subject to execute instructions contained in the Object.

Access Control Lists (ACLs) 1412 are made up of Entries 1414. Each entry two components: Subject Template 1416 and Mode Specifier 1418. Subject Template 1416 specifies a group of subjects that may reference the Object and Mode Specifier 1418 specifies the kinds of access these subjects may have to the Object. Logically speaking, ACL 1412 is checked each time a process references an Object 1410. The reference may succeed only if the process's current Subject 1408 is one of those on Object 1410's ACL 1412 and if the modes in the ACL Entry 1414 for the Subject 1408 allow the kind of access the process wishes to make.

20 11. Virtual Processors and Virtual Processor Swapping (Fig. 15)

As previously mentioned, the execution of a program by a Process 610 cannot take place unless EOS 704 has bound the Process 610 to a Virtual Processor 612. Physical execution of the Process 610 takes place only while the process's Virtual Processor 612 is bound to JP 114. The following discussion deals with the data bases belonging to a Virtual Processor 612 and the means by which a Virtual Processor 612 is bound to and removed from JP 114.

25 Fig. 15 illustrates the devices and tables which KOS 706, 710 uses to implement Virtual Processors 612. FU 120 WCS contains KOS Microcode 706 for binding Virtual Processors 612 to JP 114 and removing them from JP 114. Timers 1502 and Interrupt Line 1504 are hardware devices which produce signals that cause the invocation of KOS Microcode 706. Timers 1502 contains two timing devices: Interval Timer 1506, which may be set by KOS 706, 710 to signal when a certain time is reached, and Egg Timer 1508, which guarantees that there is a maximum time interval for which a Virtual processor 612 can be bound to JP 114 before it invokes KOS Microcode 706. Interrupt Line 1504 becomes active when JP 114 receives a message from IOS 116, for example when IOS 116 has finished loading a logical page into MEM 112.

30 FU 120 Registers 508 contain state belonging to the Virtual Processor 612 currently bound to JP 114. Here, this Virtual Processor 612 is called Virtual Processor A. In addition, Registers 508 contain registers reserved for the execution of VP Swapping Microcode 1510. ALU 1942 (part of FU 120) is used for the descriptor-to-pointer and pointer-to-descriptor transformations required when one Virtual Processor 612 is unbound from JP 114 and another bound to JP 114. MEM 112 contains data bases for Virtual Processors 612 and data bases used by KOS 706, 710 to manage Virtual Processors 612. KOS 706, 710 provides a fixed number of Virtual Processors 612 for CS 101. Each Virtual Processor 612 is represented by a Virtual Processor State Block (VPSB) 614. Each VPSB 614 contains information used by KOS 706, 710 to manage the Virtual Processor 612, and in addition contains information associating the Virtual Processor 612 with a process. Fig. 15 shows two VPSBs 614, one belonging to Virtual Processor 612A, and another belonging to Virtual Processor 612B, which will replace Virtual Processor 612A on JP 114. The VPSBs 614 are contained in VPSB Array 1512. The index of a VPSB 614 in VPSB Array 1512 is Virtual Processor Number 1514 belonging to the Virtual Processor 612 represented by a VPSB 614. Virtual Processor Lists 1516 are lists which KOS 706, 710 uses to manage Virtual Processors 612. If a Virtual Processor 612 is able to execute, its Virtual Processor Number 1514 is on a list called the Runnable List; Virtual Processors 612 which cannot run are on other lists, depending on the reason why they cannot run. It is assumed that Virtual Processor 612B's Virtual Processor Number 1514 is the first one on the Runnable List.

35 When a process is bound to a Virtual Processor 612, the Virtual Processor Number 1514 is copied into the process's Process Object 901 and the AONs of the process's Process Object 901 and stacks are copied into the Virtual Processor 612's VPSB 614. (AONs are used because a process's stacks are wired active as long as the process is bound to a Virtual Processor 612). Binding is carried out by KOS 706, 710 at the request of EOS 704. In Fig. 15, two Secure Stack Objects 906 are shown, one belonging to the process to which Virtual Processor 612A is bound, and one belonging to that to which Virtual Processor 612B is bound.

45 Having described certain overall operating features of CS 101, a present implementation of CS 101's structure will be described further next below.

60 E. CS 101 Structural Implementation (Figs. 16, 17, 18, 19, 20)

1. (IOS) 116 (Figs. 16, 17)

Referring to Fig. 16, a partial block diagram of IOS 116 is shown. Major elements of IOS 116 include an ECLIPSE® Burst Multiplexer Channel (BMC) 1614 and a NOVA® Data Channel (NDC) 1616, an IO Controller (IOC) 1618 and a Data Mover (DM) 1610. IOS 116's data channel devices, for example BMC 1614 and NDC 65 1616, comprise IOS 116's interface to the outside world. Information and addresses are received from

external devices, such as disk drives, communications modes, or other computer systems, by IOS 116's data channel devices and are transferred to DM 1610 (described below) to be written into MEM 112. Similarly, information read from MEM 112 is provided through DM 1610 to IOS 116's data channel devices and thus to the above described external devices. These external devices are a part of CS 101's addressable memory space and may be addressed through UID addresses.

IOC 1618 is a general purpose CPU, for example an ECLIPSE® computer available from Data General Corporation. A primary function of IOC 1618 is control of data transfer through IOS116. In addition, IOC 1618 generates individual Maps for each data channel device for translating external device addresses into physical addresses within MEM 112. As indicated in Fig. 16, each data channel device contains an individual Address Translation Map (MAP) 1632 and 1636. This allows IOS 116 to assign individual areas of MEM 112's physical address space to each data channel device. This feature provides protection against one data channel device writing into or reading from information belonging to another data channel device. In addition, IOC 1618 may generate overlapping address translation Maps for two or more data channel devices to allow these data channel devices to share a common area of MEM 112 physical address space.

Data transfer between IOS 116's data channel devices and MEM 112 is through DM 1610, which includes a Buffer memory (BUF) 1641. BUF 1641 allows MEM 112 and IOS 116 to operate asynchronously. DM 1610 also includes a Ring Grant Generator (RGG) 1644 which controls access of various data channel devices to MEM 112. RGG 1644 is designed to be flexible in apportioning access to MEM 112 among IOS 116's data channel devices as loads carried by various data channel devices varies. In addition, RGG 1644 insures that no one, or group, of data channel devices may monopolize access to MEM 112.

Referring to Fig. 17, a diagramic representation of RGG 1644's operation is shown. As described further in a following description, RGG 1644 may be regarded as a commutator scanning a number of ports which are assigned to various channel devices. For example, ports A, C, E, and G may be assigned to a BMC 1614, ports B and F to a NDC 1616, and ports D and H to another data channel device. RGG 1644 will scan each of these ports in turn and, if the data channel device associated with a particular port is requesting access to MEM 112, will grant access to MEM 112 to that data channel device. If no request is present at a given port, RGG 1644 will continue immediately to the next port. Each data channel device assigned one or more ports is thereby insured opportunity of access to MEM 112. Unused ports, for example indicating data channel devices which are not presently engaged in information transfer, are effectively skipped over so that access to MEM 112 is dynamically modified according to the information transfer loads of the various data channel devices. RGG 1644's ports may be reassigned among IOS 116's various data channel devices as required to suit the needs of a particular CS 101 system. If, for example, a particular CS 101 utilizes NDC 1616 more than a BMC 1614, that CS 101's NDC 1616 may be assigned more ports while that CS 101's BMC 1614 is assigned fewer ports.

2. Memory (MEM) 112 (Fig. 18)

Referring to Fig. 18, a partial block diagram of MEM 112 is shown. Major elements of MEM 112 are Main Store Bank (MSB) 1810, a Bank Controller (BC) 1814, a Memory Cache (MC) 1816, a Field Interface Unit (FIU) 1820, and Memory Interface Controller (MIC) 1822. Interconnections of these elements with input and output buses of MEM 112 to IOS 116 and JP 114 are indicated.

MEM 112 is an intelligent, prioritizing memory having a single port to IOS 116, comprised of IOM Bus 130, MIO Bus 129, and IOMC Bus 131, and dual ports to JP 114. A first JP 114 port is comprised of MOD Bus 140 and PD Bus 146, and a second port is comprised of JPD Bus 142 and PD Bus 146. In general, all data transfers from and to MEM 112 by IOS 116 and JP 114 are of single, 32 bit words; IOM Bus 130, MIO Bus 129, MOD Bus 140, and JPD Bus 142 are each 32 bits wide. CS 101, however, is a variable word length machine wherein the actual physical width of data buses are not apparent to a user. For example, a Name in a user's program may refer to an operand containing 97 bits of data. To the user, that 97 bit data item will appear to be read from MEM 112 to JP 114 in a single operation. In actuality, JP 114 will read that operand from MEM 112 in a series of read operations referred to as a string transfer. In this example, the string transfer will comprise three 32 bit read transfers and one single bit read transfer. The final single bit transfer, containing a single data bit, will be of a 32 bit word wherein one bit is data and 31 bits are fill. Write operations to MEM 112 may be performed in the same manner. If a single read or write request to MEM 112 specifies a data item of less than 32 bits of data, that transfer will be accomplished in the same manner as the final transfer described above. That is, a single 32 bit word will be transferred wherein non-data bits are fill bits.

Bulk data storage in MEM 112 is provided in MSB 1810, which is comprised of one or more Memory Array cards (MAs) 1812. The data path into and out of MA 1812 is through BC 1814, which performs all control and timing functions for MAs 1812. BC 1814's functions include addressing, transfer of data, controlling whether a read or write operation is performed, refresh, sniffing, and error correction code operations. All read and write operations from and to MAs 1812 through BC 1814 are in blocks of four 32 bit words.

The various MAs 1812 comprising MSB 1810 need not be of the same data storage capacity. For example, certain MAs 1812 may have a capacity of 256 kilobytes while other MAs 1812 may have a capacity of 512 kilobytes. Addressing of the MAs 1812 in MSB 1810 is automatically adapted to various MA 1812

configurations. As indicated in Fig. 18, each MA 1812 contains an address circuit (A) which receives an input from the next lower MA 1812 indicating the highest address in that next lower MA 1812. The A circuit on an MA 1812 also receives an input from that MA 1812 indicating the total address space of that MA 1812. The A circuit of that MA 1812 adds the highest address input from next lower MA 1812 to its own input representing its own capacity and generates an output to the next MA 1812 indicating its own highest address. All MAs 1812 of MSB 1810 are addressed in parallel by BC 1814. Each MA 1812 compares such addresses to its input from the next lower MA 1812, representing highest address of that next lower MA 1812, and its own output, representing its own highest address, to determine whether a particular address provided by BC 1814 lies within the range of addresses contained within that particular MA 1812. The particular MA 1812 whose address space includes that address will then respond by accepting the read or write request from BC 1814.

MC 1816 is the data path for transfer of data between BC 1814 and IOS 116 and JP 114. MC 1816 contains a high speed cache storing data from MSB 1810 which is currently being utilized by either IOS 116 or JP 114. MSB 1810 thereby provides MEM 112 with a large storage capacity while MC 1816 provides the appearance of a high speed memory. In addition to operating as a cache, MC 1816 includes a bypass write path which allows IOS 116 to write blocks of four 32 bit words directly into MSB 1810 through BC 1814. In addition, MC 1816 includes a cache write-back path which allows data to be transferred out of MC 1816's cache and stored while further data is transferred into MC 1816's cache. Displaced data from MC 1816's cache may then be written back into MSB 1810 at a later, more convenient time. This write-back path enhances speed of operation of MC 1816 by avoiding delays incurred by transferring data from MC 1816 to MSB 1810 before new data may be written into MC 1816.

MEM 112's FIU 1820 allows manipulation of data formats in writes to and reads from MEM 112 by both JP 114 and IOS 116. For example, FIU 1820 may convert unpacked decimal data to packed decimal data, and vice versa. In addition, FIU 1820 allows MEM 112 to operate as a bit addressable memory. For example, as described all data transfers to and from MEM 112 are of 32 bit words. If a data transfer of less than 32 bits is required, the 32 bit word containing those data bits may be read from MC 1816 to FIU 1820 and therein manipulated to extract the required data bits. FIU 1820 then generates a 32 bit word containing those required data bits, plus fill bits, and provides that new 32 bit word to JP 114 or IOS 116. When writing into MEM 112 from IOS 116 through FIU 1820, data is transferred onto IOM Bus 130, read into FIU 1820, operated upon, transferred onto MOD Bus 140, and transferred from MOD Bus 140 to MC 1816. In read operations from MEM 112 to IOS 116, data is transferred from MC 1816 to MOD Bus 140, written into FIU 1820 and operated upon, and transferred onto MIO Bus 129 to IOS 116. In a data read from MEM 112 to JP 114, data is transferred from MC 1816 onto MOD Bus 140, transferred into FIU 1820 and operated upon, and transferred again onto MOD Bus 140 to JP 114. In write operations from JP 114 to MEM 112, data on JPD Bus 142 is transferred into FIU 1820 and operated upon, and is then transferred onto MOD Bus 140 to MC 1816. MOD Bus 140 is thereby utilized as an MEM 112 internal bus for FIU 1820 operations.

Finally, MIC 1822 provides primary control of BC 1814, MC 1816, and FIU 1820. MIC 1822 receives control inputs from and provides control outputs to PD Bus 146 and IOMC Bus 131. MIC 1822 contains primary microcode control for MEM 112, but BC 1814, MC 1816, and FIU 1820 each include internal microcode control. Independent, internal microcode controls allow BC 1814, MC 1816, and FIU 1820 to operate independently of MIC 1822 after their operations have been initiated by MIC 1822. This allows BC 1814 and MSB 1810, MC 1816, and FIU 1820 to operate independently and asynchronously. Efficiency and speed of operation of MEM 112 are thereby enhanced by allowing pipelining of MEM 112 operations.

3. Fetch Unit (FU) 120 (Fig. 19)

A primary function of FU 120 is to execute SInS. In doing so, FU 120 fetches instructions and data (SOPs and Names) from MEM 112, returns results of operations to MEM 112, directs operation of EU 122, executes instructions of user's programs, and performs the various functions of CS 101's operating systems. As part of these functions, FU 120 generates and manipulates logical addresses and descriptors and is capable of operating as a general purpose CPU.

Referring to Fig. 19, a major element of FU 120 is the Descriptor Processor (DESP) 1910. DESP 1910 includes General Register File (GRF) 506. GRF 506 is a large register array divided vertically into three parts which are addressed in parallel. A first part, AONGRF 1932, stores AON fields of logical addresses and descriptors. A second part, OFFGRF 1934, stores offset fields of logical addresses and descriptors and is utilized as a 32 bit wide general register array. A third portion GRF 506, LENGRF 1936, is a 32 bit wide register array for storing length fields of logical descriptors and as a general register for storing data. Primary data path from MEM 112 to FU 120 is through MOD Bus 140, which provides inputs to OFFGRF 1934. As indicated in Fig. 19, data may be transferred from OFFGRF 1934 to inputs of AONGRF 1932 and LENGRF 1936 through various interconnections. Similarly, outputs from LENGRF 1936 and AONGRF 1932 may be transferred to inputs of AONGRF 1932, OFFGRF 1934, and LENGRF 1936.

Output of OFFGRF 1934 is connected to inputs of DESP 1910's Arithmetic and Logic Unit (ALU) 1942. ALU 1942 is a general purpose 32 bit ALU which may be used in generating and manipulating logical addresses and descriptors, as distinct from general purpose arithmetic and logic operands performed by MUX 1940. Output of ALU 1942 is connected to JPD Bus 142 to allow results of arithmetic and logic operations to be transferred to MEM 112 or EU 122.

EP 0 067 556 B1

Also connected from output of OFFGRF 1934 is Descriptor Multiplexer (MUX) 1940. An output of MUX 1940 is provided to an input of ALU 1942. MUX 1940 is a 32 bit ALU, including an accumulator, for data manipulation operations. MUX 1940, together with ALU 1942, allows DESP 1910 to perform 32 bit arithmetic and logic operations. MUX 1940 and ALU 1942 may allow arithmetic and logic operations upon
5 operands of greater than 32 bits by performing successive operations upon successive 32 bit words of larger operands.

Logical descriptors or addresses generated or provided by DESP 1910, are provided to Logical Descriptor (LD) Bus 1902. LD Bus 1902 in turn is connected to an input of Address Translation Unit (ATU) 1928. ATU 1928 is a cache mechanism for converting logical descriptors to MEM 112 physical descriptors.

10 LD Bus 1902 is also connected to write input of Name Cache (NC) 1926. NC 1926 is a cache mechanism for storing logical descriptors corresponding to operand Names currently being used in user's programs. As previously described, Name Table Entries corresponding to operands currently being used in user's programs are stored in MEM 112. Certain Name Table Entries for operands of a user's program currently being executed are transferred from those Name Tables in MEM 112 to FU 120 and are therein evaluated to
15 generate corresponding logical descriptors. These logical descriptors are then stored in NC 1926. As will be described further below, the instruction stream of a user's program is provided to FU 120's Instruction Buffer (IB) 1962 through MOD Bus 140. FU 120's Parser (P) 1964 separates out, or parses, Names from IB 1962 and provides those Names as address inputs to NC 1924. NC 1924 in turn provides logical descriptor outputs to LD Bus 1902, and thus to input of ATU 1928. NC 1926 input from LD Bus 1902 allows logical
20 descriptors resulting from evaluation of Name Table Entries to be written into NC 1926. FU 120's Protections Cache (PC) 1934 is a cache mechanism having an input connected from LD Bus 1902 and providing information, as described further below, regarding protection aspects of references to data in MEM 112 by user's programs. NC 1926, ATU 1928, and PC 1934 are thereby acceleration mechanisms of, respectively, CS 101's Namespace addressing, logical to physical address structure, and protection
25 mechanism.

Referring again to DESP 1910, DESP 1910 includes BIAS 1952, connected from output of LENGRF 1936. As previously described, operands containing more than 32 data bits are transferred between MEM 112 and JP 114 by means of string transfers. In order to perform string transfers, it is necessary for FU 120 to generate a corresponding succession of logical descriptors wherein length fields of those logical
30 descriptors is no greater than 5 bits, that is, specify lengths of no greater than 32 data bits.

A logical descriptor describing a data item to be transferred by means of a string transfer will be stored in GRF 506. AON field of the logical descriptor will reside in AONGRF 1932, O field in OFFGRF 1934, and L field in LENGRF 1936. At each successive transfer of a 32 bit word in the string transfer, O field of that original logical descriptor will be incremented by the number of data bits transferred while L field will be
35 accordingly decremented. The logical descriptor residing in GRF 506 will thereby describe, upon each successive transfer of the string transfer, that portion of the data item yet to be transferred. O field in OFFGRF 1934 will indicate increasingly larger offsets into that data item, while L field will indicate successively shorter lengths. AON and O fields of the successive logical descriptors of the string transfer. L field of the logical
40 descriptor residing in LENGRF 1936, however, may not be so used as L fields of the successive string transfer logical descriptors as this L field indicates remaining length of data item yet to be transferred. Instead, BIAS 1952 generates the 5 bit L fields of successive string transfer logical descriptors while correspondingly decrementing L field of the logical descriptor in LENGRF 1936. During each transfer, BIAS
45 1952 generates L field of the *next* string transfer logical descriptor while concurrently providing L field of the *current* string transfer logical descriptor. By doing so, BIAS 1952 thereby increases speed of execution of string transfers by performing pipelined L field operations. BIAS 1952 thereby allows CS 101 to appear to the user to be a variable word length machine by automatically performing string transfers. This mechanism is used for transfer of any data item greater than 32 bits, for example double precision floating
50 point numbers.

Finally, FU 120 includes microcode circuitry for controlling all FU 120 operations described above. In particular, FU 120 includes a microinstruction sequence control store (mC) 1920 storing sequences of microinstructions for controlling step by step execution of all FU 120 operations. In general, these FU 120 operations fall into two classes. A first class includes those microinstruction sequences directly concerned with executing the SOPs of user's programs. The second class includes microinstruction sequences
55 concerned with CS 101's operating systems, including and certain automatic, internal FU 120 functions such as evaluation of Name Table Entries.

As previously described, CS 101 is a multiple S-Language machine. For example, mC 1920 may contain microinstruction sequences for executing user's SOPs in at least four different Dialects. mC 1920 is comprised of a writeable control store and sets of microinstruction sequences for various Dialects may be
60 transferred into and out of mC 1920 as required for execution of various user's programs. By storing sets of microinstruction sequences for more than one Dialect in mC 1920, it is possible for user's programs to be written in a mixture of user languages. For example, a particular user's program may be written primarily in FORTRAN but may call certain COBOL routines. These COBOL routines will be correspondingly translated into COBOL dialect SOPs and executed by COBOL microinstruction sequences stored in mC 1920.

65 The instruction stream provided to FU 120 from MEM 112 has been previously described with

EP 0 067 556 B1

reference to Fig. 3. SOPs and Names of this instruction stream are transferred from MOD Bus 140 into IB 1962 as they are provided from MEM 112. IB 1962 includes two 32 bit (one word) registers. IB 1962 also includes prefetch circuitry for reading for SOPs and Names of the instruction stream from MEM 112 in such a manner that IB 1962 shall always contain at least one SOPs or Name. FU 120 includes (P) 1964 which reads and separates, or parses, SOPs and Names from IB 1962. As previously described, P 1964 provides those Names to NC 1926, which accordingly provides logical descriptors to ATU 1928 so as to read the corresponding operands from MEM 112.

SOPs parsed by P 1964 are provided as inputs to Fetch Unit Dispatch Table (FUDT) 1904 and Execute Unit Dispatch Table (EUDT) 1966. Referring first to FUDT 1904, FUDT 1904 is effectively a table for translating SOPs to starting addresses in mC 1912 of corresponding microinstruction sequences. This intermediate translation of SOPs to mC 1912 addresses allows efficient packing of microinstruction sequences within mC 1912. That is, certain microinstruction sequences may be common to two or more S-Language Dialects. Such microinstruction sequences may therefore be written into mC 1912 once and may be referred to by different SOPs of different S-Language Dialects.

EUDT 1966 performs a similar function with respect to EU 122. As will be described below, EU 122 contains a mC, similar to mC 1912, which is addressed through EUDT 1966 by SOPs specifying EU 122 operations. In addition, FU 120 may provide such addresses mC 1912 to initiate EU 122 operations as required to assist certain FU 120 operations. Examples of such operations which may be requested by FU 120 include calculations required in evaluating Name Table Entries to provide logical descriptors to be loaded into NC 1926.

Associated with both FUDT 1904 and EUDT 1966 are Dialect (D) registers 1905 and 1967. D registers 1905 and 1967 store information indicating the particular S-Language Dialect currently being utilized in execution of a user's program. Outputs of D registers 1905 and 1967 are utilized as part of the address inputs to mC 1912 and EU 122's mC.

4. Execute Unit (EU) 122 (Fig. 20)

As previously described, EU 122 is an arithmetic and logic unit provided to relieve FU 120 of certain arithmetic operations. EU 122 is capable of performing addition, subtraction, multiplication, and division operations on integer, packed and unpacked decimal, and single and double precision floating point operands. EU 122 is an independently operating microcode controlled machine including Microcode Control (mC) 2010 which, as described above, is addressed by EUDT 1966 to initiate EU 122 operations. mC 2010 also includes logic for handling mutual interrupts between FU 120 and EU 122. That is, FU 120 may interrupt current EU 122 operations to call upon EU 122 to assist an FU 120 operation. For example, FU 120 may interrupt an arithmetic operation currently being executed by EU 122 to call upon EU 122 to assist in generating a logical descriptor from a Name Table Entry.

Similarly, EU 122 may interrupt current FU 120 operations when EU 122 requires FU 120 assistance in executing a current arithmetic operation. For example, EU 122 may interrupt a current FU 120 operation if EU 122 receives an instruction and operands requiring EU 122 to perform a divide by zero.

Referring to Fig. 20, a partial block diagram of EU 122 is shown. EU 122 includes two arithmetic and logic units. A first arithmetic and logic unit (MULT) 2014 is utilized to perform addition, subtraction, multiplication, and division operations upon integer and decimal operands, and upon mantissa fields of single and double precision floating point operands. Second ALU (EXP) 2016 is utilized to perform operations upon single and double precision floating point operand exponent fields in parallel with operations performed upon floating point mantissa fields by MULT 2014. Both MULT 2014 and EXP 2016 include an arithmetic and logic unit, respectively MALU 2074 and EXPALU 2084. MULT 2014 and EXP 2016 also include register files, respectively MRF 2050 and ERF 2080, which operate and are addressed in parallel in a manner similar to AONGRF 1932, OFFGRF 1984 and LENGRF 1936.

Operands for EU 122 to operate upon are provided from MEM 112 through MOD Bus 140 and are transferred into Operand Buffer (OPB) 2022. In addition to serving as an input buffer, OPB 2022 performs certain data format manipulation operations to transform input operands into formats most efficiently operated with by EU 122. In particular, EU 122 and MULT 2014 may be designed to operate efficiently with packed decimal operands. OPB 2022 may transform unpacked decimal operands into packed decimal operands. Unpacked decimal operands are in the form of ASCII characters wherein four bits of each character are binary codes specifying a decimal value between zero and nine. Other bits of each character are referred to as zone fields and in general contain information identifying particular ASCII characters. For example, zone field bits may specify whether a particular ASCII character is a number, a letter, or punctuation. Packed decimal operands are comprised of a series of four bit fields wherein each field contains a binary number specifying a decimal value of between zero and nine. OPB 2022 converts unpacked decimal to packed decimal operands by extracting zone field bits and packing the four numeric value bits of each character into the four bit fields of a packed decimal number.

EU 122 is also capable of transforming the results of arithmetic operations, for example in packed decimal format, into unpacked decimal format for transfer back to MEM 112 or FU 120. In this case, a packed decimal result appearing at output of MALU 2074 is written into MRF 2050 through a multiplexer, not shown in Fig. 20, which transforms the four bit numeric code fields of the packed decimal results into corresponding bits of unpacked decimal operand characters, and forces blanks into the zone field bits of

those unpacked decimal characters. The results of this operation are then read from MRF 2050 to MALU 2074 and zone field bits for those unpacked decimal characters are read from Constant Store (CST) 2060 to MALU 2074. These inputs from MRF 2050 and CST 2060 are added by MALU 2074 to generate final result outputs in unpacked decimal format. These final results may then be transferred onto JPD Bus 142 through Output Multiplexer (OM) 2024.

Considering first floating point operations, in addition or subtraction of floating point operands it is necessary to equalize the values of the floating point operand exponent fields. This is referred to as prealignment. In floating point operations, exponent fields of the two operands are transferred into EXPALU 2034 and compared to determine the difference between exponent fields. An output representing difference between exponent fields is provided from EXPALU 2034 to an input of floating point control (FPC) 2002. FPC 2002 in turn provides control outputs to MALU 2074, which has received the mantissa fields of the two operands. MALU 2074, operating under direction of FPC 2002, accordingly right or left shifts one operand's mantissa field to effectively align that operand's exponent field with the other operand's exponent field. Addition or subtraction of the operand's mantissa fields may then proceed.

EXPALU 2034 also performs addition or subtraction of floating point operand exponent fields in multiplication or division operations, while MALU 2074 performs multiplication and division of the operand mantissa fields. Multiplication and division of floating point operand mantissa fields by MALU 2074 is performed by successive shifting of one operand, corresponding generation of partial products of the other operand, and successive addition and subtraction of those partial products.

Finally, EU 122 performs normalization of the results of floating point operand operations by left shifting of a final result's mantissa field to eliminate zeros in the most significant characters of the final result mantissa field, and corresponding shifting of the final result exponent fields. Normalization of floating point operation results is controlled by FPC 2002. FPC 2002 examines an unnormalized floating point result output of MALU 2074 to detect which, if any, of the most significant characters of that result contain zeros. FPC 2002 then accordingly provides control outputs to EXPALU 2034 and MALU 2074 to correspondingly shift the exponent and mantissa fields of those results so as to eliminate leading character zeros from the mantissa field. Normalized mantissa and exponent fields of floating point results may then be transferred from MALU 2074 and EXPALU 2034 to JPD Bus 142 through OM 2024.

As described above, EU 122 also performs addition, subtraction, multiplication, and division operations on operands. In this respect, EU 122 uses a leading zero detector in FPC 2002 in efficiently performing multiplication and division operations. FPC 2002's leading zero detector examines the characters or bits of two operands to be multiplied or divided, starting from the highest, to determine which, if any, contain zeros so as not to require a multiplication or division operation. FPC 2002 accordingly left shifts the operands to effectively eliminate those characters or bits, thus reducing the number of operations to multiply or divide the operands and accordingly reducing the time required to operate upon the operands.

Finally, EU 122 utilizes a unique method, with associated hardware, for performing arithmetic operations on decimal operands by utilizing circuitry which is otherwise conventionally used only to perform operations upon floating point operands. As described above, MULT 2074 is designed to operate with packed decimal operands, that is operands in the form of consecutive blocks of four bits wherein each block of four bits contains a binary code representing numeric values of between zero and nine. Floating point operands are similarly in the form of consecutive blocks of four bits. Each block of four bits in a floating point operand, however, contains a binary number representing a hexadecimal value of between zero and fifteen. As an initial step in operating with packed decimal operands, those operands are loaded, one at a time, into MALU 2074 and, with each such operand, a number comprised of all hexadecimal sixes is loaded into MALU 2074 from CST 2060. This CST 2060 number is added to each packed decimal operand to effectively convert those packed decimal operands into hexadecimal operands wherein the four bit blocks contain numeric values in the range of six to fifteen, rather than in the original range of zero to nine. MULT 2014 then performs arithmetic operation upon those transformed operands, and in doing so detects and saves information regarding which four bit characters of those operands have resulted in generation of carries during the arithmetic operations. In a final step, the intermediate result resulting from completion of those arithmetic operations upon those transformed operands are reconverted to packed decimal format by subtraction of hexadecimal sixes from those characters for which carries have been generated. Effectively, EU 122 converts packed decimal operands into "Excess Six" operands, performs arithmetic operations upon those "Excess Six" operands, and reconverts "Excess Six" results of those operations back into packed decimal format.

Finally, as previously described FU 120 controls transfer of arithmetic results from EU 122 to MEM 112. In doing so, FU 120 generates a logical descriptor describing the size of MEM 112 address space, or "container", that result is to be transferred into. In certain arithmetic operations, for example integer operations, an arithmetic result may be larger than anticipated and may contain more bits than the MEM 112 "container". Container Size Check Circuit (CSC) 2052 compares actual size of arithmetic results and L fields of MEM 112 "container" logical descriptors. CSC 2052 generates an output indicating whether an MEM 112 "container" is smaller than an arithmetic result.

Having briefly described certain features of CS 101 structure and operation in the above overview, these and other features of CS 101 will be described in further detail next below in a more detailed

EP 0 067 556 B1

introduction of CS 101 structure and operation. Then, in further descriptions, these and other features of CS 101 structure and operation will be described in depth.

1. Introduction (Figs. 101—110)

A. General Structure and Operation (Fig 101)

a. General Structure

Referring to Fig. 101, a partial block diagram of Computer System (CS) 10110 is shown. Major elements of CS 10110 are Dual Port Memory (MEM) 10112, Job Processor (JP) 10114, Input/Output System (IOS) 10116, and Diagnostic Processor (DP) 10118. JP 10114 includes Fetch Unit (FU) 10120 and Execute Unit (EU) 10122.

Referring first to IOS 10116, IOS 10116 is interconnected with External Devices (ED) 10124 through Input/Output (I/O) Bus 10126. ED 10124 may include, for example, other computer systems, keyboard/display units, and disc drive memories. IOS 10116 is interconnected with Memory Input/Output (MIO) Port 10128 of MEM 10112 through Input/Output to Memory (IOM) Bus 10130 and Memory to Input/Output (MIO) Bus 10129, and with FU 10120 through I/O Job Processor (IOJP) Bus 10132.

DP 10118 is interconnected with, for example, external keyboard/CRT Display Unit (DU) 10134 through Diagnostic Processor Input/Output (DPIO) Bus 10136. DP 10118 is interconnected with IOS 10116, MEM 10112, FU 10120, and EU 10122 through Diagnostic Processor (DP) Bus 10138.

Memory to Job Processor (MJP) Port 10140 of Memory 10112 is interconnected with FU 10120 and EU 10122 through Job Processor Data (JPD) Bus 10142. An output of MJP 10140 is connected to inputs of FU 10120 and EU 10122 through Memory Output Data (MOD) Bus 10144. An output of FU 10120 is connected to an input of MJP 10140 through Physical Descriptor (PD) Bus 10146. FU 10120 and EU 10122 are interconnected through Fetch/Execute (F/E) Bus 10148.

b. General Operation

As will be discussed further below, IOS 10116 and MEM 10112 operate independently under general control of JP 10114 in executing multiple user's programs. In this regard, MEM 10112 is an intelligent, prioritizing memory having separate and independent ports MIO 10128 and MJP 10140 to IOS 10116 and JP 10114 respectively. MEM 10112 is the primary path for information transfer between External Devices 10124 (through IOS 10116) and JP 10114. MEM 10112 thus operates both as a buffer for receiving and storing various individual user's programs (e.g., data, instructions, and results of program execution) and as a main memory for JP 10114.

A primary function of IOS 10116 is as an input/output buffer between CS 10110 and ED 10124. Data and instructions are transferred from ED 10124 to IOS 10116 through I/O Bus 10126 in a manner and format compatible with ED 10124. IOS 10116 receives and stores this information, and manipulates the information into formats suitable for transfer into MEM 10112. IOS 10116 then indicates to MEM 10112 that new information is available for transfer into MEM 10112. Upon acknowledgement by MEM 10112, this information is transferred into MEM 10112 through IOM Bus 10130 and MIO Port 10128. MEM 10112 stores the information in selected portions of MEM 10112 physical address space. At this time, IOS 10116 notifies JP 10114 that new information is present in MEM 10112 by providing a "semaphore" signal to FU 10120 through IOJP Bus 10132. As will be described further below, CS 10110 manipulates the data and instructions stored in MEM 10112 into certain information structures used in executing user's programs. Among these structures are certain structures, discussed further below, which are used by CS 10110 in organizing and controlling flow and execution of user programs.

FU 10120 and EU 10122 are independently operating microcode controlled "machines" together comprising the CS 10110 micromachine for executing user's programs stored in MEM 10112. Among the principal functions of FU 10120 are: (1) fetching and interpreting instructions and data from MEM 10112 for use by FU 10120 and EU 10122; (2) organizing and controlling flow of user programs; (3) initiating EU 10122 operations; (4) performing arithmetic and logic operations on data; (5) controlling transfer of data from FU 10120 and EU 10122 to MEM 10112; and, (6) maintaining certain "stack" and "register" mechanisms, described below. FU 10120 "cache" mechanisms, also described below, are provided to enhance the speed of operation of JP 10114. These cache mechanisms are acceleration circuitry including, in part, high speed memories for storing copies of selected information stored in MEM 10112. The information stored in this acceleration circuitry is therefore more rapidly available to JP 10114. EU 10122 is an arithmetic unit capable of executing integer, decimal, or floating point arithmetic operations. The primary function of EU 10122 is to relieve FU 10120 from certain extensive arithmetic operations, thus enhancing the efficiency of CS 10110.

In general, operations in JP 10114 are executed on a memory to memory basis; data is read from MEM 10112, operated upon, and the results returned to MEM 10112. In this regard, certain stack and cache mechanisms in JP 10114 (described below) operate as extensions of MEM 10112 address space.

In operation, FU 10120 reads data and instructions from MEM 10112 by providing physical addresses to MEM 10112 by way of PA Bus 10146 and MJP Port 10140. The instructions and data are transferred to FU 10120 and EU 10122 by way of MJP Port 10140 and MOD Bus 10144. Instructions are interpreted by FU 10120 microcode circuitry, not shown in Fig. 101 but described below, and when necessary, microcode instructions are provided to EU 10122 from FU 10120's microcode control by way of F/E Bus 10148, or by way of JPD Bus 10142.

EP 0.067 556 B1

As stated above, FU 10120 and EU 10122 operate asynchronously with respect to each other's functions. A microinstruction from FU 10120 microcode circuitry to EU 10122 may initiate a selected operation of EU 10122. EU 10122 may then proceed to independently execute the selected operation. FU 10120 may proceed to concurrently execute other operations while EU 10122 is completing the selected arithmetic operation. At completion of the selected arithmetic operation, EU 10122 signals FU 10120 that the operation results are available by way of a "handshake" signal through F/E Bus 10148. FU 10120 may then receive the arithmetic operation results for further processing or, as discussed momentarily, may directly transfer the arithmetic operation results to MEM 10112. As described further below, an instruction buffer referred to as a "queue" between FU 10120 and EU 10122 allows FU 10120 to assign a sequence of arithmetic operations to be performed by EU 10122.

Information, such as results of executing an instruction, is written into MEM 10112 from FU 10120 or EU 10122 by way of JPD Bus 10142. FU 10120 provides a "physical write address" signal to MEM 10112 by way of PA Bus 10146 and MJP Port 10140. Concurrently, the information to be written into MEM 10112 is placed on JPD Bus 10142 and is subsequently written into MEM 10112 at the locations selected by the physical write address.

FU 10120 places a semaphore signal on IOJP Bus 10132 to signal to IOS 10116 that information, such as the results of executing a user's program, is available to be read out of CS 10110. IOS 10116 may then transfer the information from MEM 10112 to IOS 10116 by way of MIO Port 10128 and IOM Bus 10130. Information stored in IOS 10116 is then transferred to ED 10124 through I/O Bus 10126.

During execution of a user's program, certain information required by JP 10116 may not be available in MEM 10112. In such cases as further described in a following discussion, JP 10114 may write a request for information into MEM 10112 and notify IOS 10116, by way of IOJP Bus 10132, that such a request has been made. IOS 10116 will then read the request and transfer the desired information from ED 10124 into MEM 10112 through IOS 10116 in the manner described above. In such operations, IOS 10116 and JP 10114 operate together as a memory manager wherein the memory space addressable by JP 10114 is termed virtual memory space, and includes both MEM 10112 memory space and all external devices to which IOS 10116 has access.

As previously described, DP 10118 provides a second interface between Computer System 10110 and the external world by way of DPIO Bus 10136. DP 10118 allows DU 10134, for example a CRT and keyboard unit or a teletype, to perform all functions which are conventionally provided by a hard (i.e., switches and lights) console. For example, DP 10118 allows DU 10134 to exercise control of Computer System 10110 for such purposes as system initialization and start up, execution of diagnostic processes, and fault monitoring and identification. DP 10118 has read and write access to most memory and register portions within each of IOS 10116, MEM 10112, FU 10120, and EU 10122 by way of DP Bus 10138. Memories and registers in CS 10110 can therefore be directly loaded or initialized during system start up, and can be directly read or loaded with test and diagnostic signals for fault monitoring and identification. In addition, as described further below, microinstructions may be loaded into JP 10114's microcode circuitry at system start up or as required.

Having described the general structure and operation of Computer System 10110, certain features of Computer System 10110 will next be briefly described to aid in understanding the following, more detailed descriptions of these and other features of Computer System 10110.

c. Definition of Certain Terms

Certain terms are used relating to the structure and operation of CS 10110 throughout the following discussions. Certain of these terms will be discussed and defined first, to aid in understanding the following descriptions. Other terms will be introduced in the following descriptions as required.

A *procedure* is a sequence of operational steps, or instructions, to be executed to perform some operation. A procedure may include data to be operated upon in performing the operation.

A *program* is a static group of one or more procedures. In general, programs may be classified as user programs, utility programs, and operating system programs. A user program is a group of procedures generated by and private to one particular user or a group of users interfacing with CS 10110. Utility programs are commonly available to all users; for example, a compiler comprises of a set of procedures for compiling a user language program into an S-language program. Operating system programs are groups of procedures internal to CS 10110 for allocation and control of CS 10110 resources. Operating system programs also define interfaces within CS 10110. For example, as will be discussed further below all operands in a program are referred to by "NAME". An operating system program translates operand NAME into the physical locations of the operands in MEM 10112. The NAME translation program thus defines the interface between operand NAME (name space addresses) and MEM 10112 physical addresses.

A *process* is an independent locus of control passing through physical, logical or virtual address spaces, or, more particularly, a path of execution through a series of programs (i.e., procedures). A process will generally include a user program and data plus one or more utility programs (e.g., a compiler) and operating system programs necessary to execute the user program.

An *object* is a uniquely identifiable portion of "data space" accessible to CS 10110. An object may be regarded as a container for information and may contain data or procedure information or both. An object may contain for example, an entire program, or set of procedures, or a single bit of data. Objects need not

be contiguously located in the data space accessible to CS 10110, and the information contained in an object need not be contiguously located in that object.

A *domain* is a state of operation of CS 10110 for the purposes of CS 10110's protection mechanisms. Each domain is defined by a set of procedures having access to objects within that domain for their execution. Each object has a single domain of execution in which it is executed if it is a procedure object, or used, if it is a data object. CS 10110 is said to be operating in a particular domain if it is executing a procedure having that domain of execution. Each object may belong to one or more domains; an object belongs to a domain if a procedure executing in that domain has potential access to the object. CS 10110 may, for example have four domains: User domain, Data Base Management System (DBMS) domain, Extended Operating System (EOS) domain, and Kernel Operating System (KOS) domain. User domain is the domain of execution of all user provided procedures, such as user or utility procedures. DBMS domain is the domain of execution for operating system procedures for storing, retrieving, and handling data. EOS domain is the domain of execution of operating system procedures defining and forming the user level interface with CS 10110, such as procedures for controlling an executing files, processes, and I/O operations. KOS domain is the domain of execution of the low level, secure operating system which manages and controls CS 10110's physical resources. Other embodiments of CS 10110 may have fewer or more domains than those just described. For example, DBMS procedures may be incorporated into the EOS domain or EOS domain may be divided by incorporating the I/O procedures into an I/O domain. There is no hardware enforced limitation on the number of, or boundaries between, domains in CS 10110. Certain CS 10110 hardware functions and structures are, however, dependent upon domains.

A *subject* is defined, for purposes of CS 10110's protection mechanisms, as a combination of the current principle (user), the current process being executed, and the domain the process is currently being executed in. In addition to principle, process, and domain, which are identified by UIDs, subject may include a Tag, which is a user assigned identification code used where added security is required. For a given process, principle and process are constant but the domain is determined by the procedure currently being executed. A process's associated subject is therefore variable along the path of execution of the process.

Having discussed and defined the above terms, certain features of CS 10110 will next be briefly described.

d. Multi-Program Operation

CS 10110 is capable of concurrently executing two or more programs and selecting the sequence of execution of programs to make most effective use of CS 10110's resources. This is referred to as multiprogramming. In this regard, CS 10110 may temporarily suspend execution of one program, for example when a resource or certain information required for that program is not immediately available, and proceed to execute another program until the required resource or information becomes available. For example, particular information required by a first program may not be available in MEM 10112 when called for. JP 10114 may, as discussed further below, suspend execution of the first program, transfer a request for that information to IOS 10116, and proceed to call and execute a second program. IOS 10116 would fetch the requested information from ED 10124 and transfer it into MEM 10112. At some time after IOS 10116 notifies JP 10114 that the requested information is available in MEM 10112, JP 10114 could suspend execution of the second program and resume execution of the first program.

e. Multi-Language Operation

As previously described, CS 10110 is a multiple language machine. Each program written in a high level user language, such as COBOL or FORTRAN, is compiled into a corresponding Soft (S) Language program. That is, in terms of a conventional computer system, each user level language has a corresponding machine language, classically defined as an assembly language. In contrast to classical assembly languages, S-Languages are mid-level languages wherein each command in a user's high level language is replaced by, in general, two or three S-Language instructions, referred to as S-Interpreters. CS 10110, as further described in following discussions, provides a set, or dialect, of microcode instructions (S-Interpreters) for each S-Language. S-Interpreters interpret S-Interpreters and provide corresponding sequences of microinstructions for detailed control of CS 10110. CS 10110's instruction set and operation may therefore be tailored to each user's program, regardless of the particular user language, so as to most efficiently execute the user's program. Computer System 10110 may, for example, execute programs in both FORTRAN and COBOL with comparable efficiency. In addition, a user may write a program in more than one high level user language without loss of efficiency. For example, a user may write a portion of his program in COBOL, but may wish to write certain portions in FORTRAN. In such cases, the COBOL portions would be compiled into COBOL S-Interpreters and executed with the COBOL dialect S-Interpreter. The FORTRAN portions would be compiled into FORTRAN S-Interpreters and executed with a FORTRAN dialect S-Interpreter. The present embodiment of CS 10110 utilizes a uniform format for all S-Interpreters. This feature allows simpler S-Interpreter structures and increases efficiency of S-Interpreter interpretation because it is not necessary to provide means for interpreting each dialect individually.

f. Addressing Structure

Each object created for use in, or by operation of, a CS 10110 is permanently assigned a Unique Identifier (UID). An object's UID allows that object to be uniquely identified and located at any time, regardless of which particular CS 10110 it was created by or for or where it is subsequently located. Thus each time a new object is defined, a new and unique UID is allocated, much as social security numbers are allocated to individuals. A particular piece of information contained in an object may be located by a logical address comprising the object's UID, an offset from the start of the object of the first bit of the segment, and the length (number of bits) of the information segment. Data within an object may therefore be addressed on a bit granular basis. As will be described further in following discussions, UID's are used within a CS 10110 as logical addresses, and, for example, as pointers. Logically, all addresses and pointers in CS 10110 are UID addresses and pointers. As previously described and as described below, however, short, temporary unique identifiers, valid only within JP 10114 and referred to as Active Object Numbers are used within JP 10114 to reduce the width of address buses and amount of address information handled.

An object becomes active in CS 10110 when it is transferred from backing store CED 10124 to MEM 10112 for use in executing a process. At this time, each such object is assigned an Active Object Number (AON). AONs are short unique identifiers and are related to the object's UIDs through certain CS 10110 information structures described below. AONs are used only within JP 10114 and are used in JP 10114, in place of UIDs, to reduce the required width of JP 10114's address buses and the amount of address data handled in JP 10114. As with UID logical addresses, a piece of data in an object may be addressed through a bit granular AON logical address comprising the object's AON, an offset from the start of the object of the first bit of the piece, and the length of the piece.

The transfer of logical addresses, for example pointers, between MEM 10112 (UIDA) and JP 10114 (AONs) during execution of a process requires translations between UIDs and AONs. As will be described in a later discussion, this translation is accomplished, in part, through the information structures mentioned above. Similarly, translation of logical addresses to physical addresses in MEM 10112, to physically access information stored in MEM 10112, is accomplished through CS 10110 information structures relating AON logical addresses to MEM 10112 physical addresses.

Each operand appearing in a program is assigned a Name when the program is compiled. Thereafter, all references to the operands are through their assigned Names. As will be described in detail in a later discussion, CS 10110's addressing structure includes a mechanism for recognizing Names as they appear in an instruction stream and Name Tables containing directions for resolving Names to AON logical addresses. AON logical addresses may then be evaluated, for example translated into a MEM 10112 physical address, to provide actual operands. The use of Names to identify operands in the instructions stream (process) (1) allows a complicated address to be replaced by a simple reference of uniform format; (2) does not require that an operation be directly defined by data type in the instruction stream; (3) allows repeated references to an operand to be made in an instruction stream by merely repeating the operand's Name; and, (4) allows partially completed Name to address translations to be stored in a cache to speed up operand references. The use of Names thereby substantially reduces the volume of information required in the instruction stream for operand references and increases CS 10110 speed and efficiency by performing operand references through a parallel operating, underlying mechanism.

Finally, CS 10110 address structure incorporates a set of Architectural Base Pointers (ABPs) for each process. ABPs provide an addressing framework to locate data and procedure information belonging to a process and are used, for example, in resolving Names to AON logical addresses.

g. Protection Mechanism

CS 10110's protection mechanism is constructed to prevent a user from (1) gaining access to or disrupting another user's process, including data, and (2) interfering with or otherwise subverting the operation of CS 10110. Access rights to each particular active object are dynamically granted as a function of the currently active subject. A subject is defined by a combination of the current principle (user), the current process being executed, and the domain in which the process is currently being executed. In addition to principle, process, and domain, subject may include a Tag, which is a user assigned identification code used where added security is required. For a given process, the principle and process are constant but the domain is determined by the procedure currently being executed. A process's associated subject is therefore variable along the path of execution of the process.

In a present embodiment of CS 10110, procedures having KOS domain of execution have access to objects in KOS, EOS, DBMS, and User domains; procedures having EOS domain of execution have access to objects in EOS, DBMS, and User domains; procedures having DBMS domain of execution have access to objects in DBMS and User domains; and procedures having User domain of execution have access only to objects in User domain. A user cannot, therefore, obtain access to objects in KOS domain of execution and cannot influence CS 10110's low level, secure operating system. The user's process may, however, call for execution a procedure having KOS domain of execution. At this point the process's subject is in the KOS domain and the procedure will have access to certain objects in KOS domain.

In a present embodiment of CS 10110, also described in a later discussion, each object has associated with it an Access Control List (ACL). An ACL contains an Access Control Entry (ACE) for each subject having access to that object. ACEs specify, for each subject, access rights a subject has with regard to that object.

EP 0 067 556 B1

There is normally no relationship, other than that defined by an object's ACL, between subjects and objects. CS 10110, however, supports Extended Type Objects having Extended ACLs wherein a user may specifically define which subjects have what access rights to the object.

5 In another embodiment of CS 10110, described in a following discussion, access rights are granted on a dynamic basis. In executing a process, a procedure may call a second procedure and pass an argument to the called procedure. The calling procedure will also pass selected access rights to that argument to the called procedure. The passed access rights exist only for the duration of the call.

10 In the dynamic access embodiment, access rights are granted only at the time they are required. In the ACL embodiment, access rights are granted upon object creation or upon specific request. In either embodiment, each procedure to which arguments may be passed in a cross-domain call has associated with it an Access Information Array (AIA). A procedure's AIA states what access rights a calling procedure (subject) must have before the called procedure can operate on the passed argument. CS 10110's protection mechanisms compare the calling procedure's access rights to the rights required by the called procedure. This ensures that a calling procedure may not ask a called procedure to do what the calling procedure is not allowed to do. Effectively, a calling procedure can pass to a called procedure only the access rights held by the calling procedure.

15 Having described the general structure and operation and certain features of CS 10110, those and other features of CS 10110 operation will next be described in greater detail.

20 B. Computer System 10110 Information Structures and Mechanisms (Figs. 102, 103, 104, 105)

CS 10110 contains certain information structures and mechanisms to assist in efficient execution of processes. These structures and mechanisms may be considered as falling into three general types. The first type concerns the processes themselves, i.e., procedure and data objects comprising a user's process or directly related to execution of a user's process. The second type are for management, control, and execution of processes. These structures are generally shared by all processes active in CS 10110. The third type are CS 10110 micromachine information structures and mechanisms. These structures are concerned with the eternal operation of the CS 10110 micromachine and are private to the CS 10110 micro-machine.

30 a. Introduction (Fig. 102)

Referring to Fig. 102, a pictorial representation of CS 10110 (MEM 10112, FU 10120, and EU 10122) is shown with certain information structures and mechanisms depicted therein. It should be understood that these information structures and mechanisms transcend or "cut across" the boundaries between MEM 10112, FU 10120, EU 10122, and IOS 10116. Referring to the upper portion of Fig. 103 Process Structures 10210 contains those information structures and mechanisms most closely concerned with individual processes, the first and third types of information structures described above. Process Structures 10210 reside in MEM 10112 and Virtual Processes 10212 include Virtual Processes (VP) 1 through N. Virtual Processes 10212 may contain, in a present embodiment of CS 10110, up to 256 VP's. As previously described, each VP includes certain objects particular to a single user's process, for example stack objects previously described and further described in a following description. Each VP also includes a Process Object containing certain information required to execute the process, for example pointers to other process information.

Virtual Processor State Blocks (VPSBs) 10218 include VPSBs containing certain tables and mechanisms for managing execution of VPs selected for execution by CS 10110.

45 A particular VP is bound into CS 10110 when a Virtual Process Dispatcher, described in a following discussion selects that VP as eligible for execution. The selected VPs Process Object, as previously described, is swapped into a VPSB. VPSBs 10218 may contain, for example 16 or 32 State Blocks so that CS 10110 may concurrently execute to 16 or 32 VPs. When a VP assigned to a VPSB is to be executed, the VP is swapped onto the information structures and mechanisms shown in FU 10120 and EU 10122. FU Register and Stack Mechanism (FURSM) 10214 and EU Register and Stack Mechanism (EURSM) 10216, shown respectively in FU 10120 and EU 10122, comprise register and stack mechanisms used in execution of VPs bound to CS 10110. These register and stack mechanisms, as will be discussed below, are also used for certain CS 10110 process management functions. Procedure Objects (POs) 10213 contains Procedure Objects (POs) 1 to N of the processes executing in CS 10110.

50 Addressing Mechanisms (AM) 10220 are a part of CS 10110's process management system and are generally associated with Computer System 10110 addressing functions as described in following discussions. UID/AON Tables 10222 is a structure for relating UID's and AON's, previously discussed. Memory Management Tables 10224 includes structures for (1) relating AON logical addresses and MEM 10112 physical addresses; (2) managing MEM 10112's physical address space; (3) managing transfer of information between MEM 10112 and CS 10110's backing store (ED 10124) and, (4) activating objects into CS 10110; Name Cache (NC) 10226 and Address Translation Cache (ATC) 10228 are acceleration mechanisms for storing addressing information relating to the VP currently bound to CS 10110. NC 10226, described further below, contains information relating operand Names to AON addresses. ATC 10228, also discussed further below, contains information relating AON addresses to MEM 10112 physical addresses.

65 Protection Mechanisms 10230, depicted below AM 10220, include protection Tables 10232 and Protection Cache (PC) 10234. Protection Tables 10232 contain information regarding access rights to each

object active in CS 10110. PC 10234 contains protection information relating to certain objects of the VP currently bound to CS 10110.

5 Microinstruction Mechanisms 10236, depicted below PM 10230, includes Micro-code (M Code) Store 10238, FU (Micro-code) M Code Structure 10240, and EU Micro-code (M Code) Structure 10242. These structures contain microinstruction mechanisms and tables for interpreting SINs and controlling the detailed operation of CS 10110. Micro-instruction Mechanisms 10232 also provide microcode tables and mechanisms used, in part, in operation of the low level, secure operating system that manages and controls CS 10110's physical resources.

10 Having thus briefly described certain CS 10110 information structures and mechanisms with the aid of Fig. 102, those information structures and mechanisms will next be described in further detail in the order mentioned above. In these descriptions it should be noted that, in representation of MEM 10112 shown in Fig. 102 and in other figures of following discussions, the addressable memory space of MEM 10112 is depicted. Certain portions of MEM 10112 address space have been designated as containing certain information structures and mechanisms. These structures and mechanisms have real physical existence in MEM 10112, but may vary in both location and volume of MEM 10112 address space they occupy. Assigning position of a single, large memory to contain these structures and mechanisms allows these structures and mechanisms to be reconfigured as required for most efficient operation of CS 10110. In an alternate embodiment, physically separate memories may be used to contain the structures and mechanisms depicted in MEM 10112, rather than assigned portions of a single memory.

20 b. Process Structure 10210 (Figs. 103, 104, 105)

Referring to Fig. 103, a partial schematic representation of Process Structures 10210 is shown. Specifically, Fig. 103 shows a Process (P) 10310 selected for execution, and its associated Procedure Objects (POs) in Process Objects (POs) 10213. P 10310 is represented in Fig. 103 as including four procedure objects in POs 10213. It is to be understood that this representation is for clarity of presentation; a particular P 10310 may include any number of procedure objects. Also for clarity of presentation, EURSM 10216 is not shown as EURSM 10216 is similar to FURSM 10214. EURSM 10216 will be described in detail in the following detailed discussions of CS 10110's structure and operation.

30 As previously discussed, each process includes certain data and procedure object. As represented in Fig. 103 for P 10310 the procedure objects reside in POs 10213. The data objects include Static Data Areas and stack mechanisms in P 10310. POs, for example KOS Procedure Object (KOSPO) 10318, contain the various procedures of the process, each procedure being a sequence of SINs defining an operation to be performed in executing the process. As will be described below, Procedure Objects also contain certain information used in executing the procedures contained therein. Static Data Areas (SDAs) are data objects generally reserved for storing data having an existence for the duration of the process. P 10310's stack mechanisms allow stacking of procedures for procedure calls and returns and for swapping processes in and out of JP 10114. Macro-Stacks (MAS) 10328 to 10334 are generally used to store automatic data (data generated during execution of a procedure and having an existence for the duration of that procedure). Although shown as separate from the stacks in P 10310, the SDAs may be contained with MASs 10328 to 40 10334. Secure Stack (SS) 10336 stores, in general, CS 10110 micro-machine state for each procedure called. Information stored in SS 10336 allows machine state to be recovered upon return from a called procedure, or when binding (swapping) a VP into CS 10110.

As shown in P 10310, each process is structured on a domain basis. A P 10310 may therefore include, for each domain, one or more procedure objects containing procedures having that domain as their domain of execution, an SDA and an MAS. For example, KOS domain of P 10310 includes KOSPO 10318, KOSSDA 10326, and KOSMAS 10334. P 10310's SS 10336 does not reside in any single domain of P 10310, but instead is a stack mechanism belonging to CS 10110 micromachine.

Having described the overall structure of a P 10310, the individual information structures and mechanisms of a P 10310 will next be described in greater detail.

50 1. Procedure Objects (Fig. 103)

KOSPO 10318 is typical of CS10110 procedure objects and will be referred to for illustration in the following discussion. Major components of KOSPO 10318 are Header 10338, External Entry Descriptor (EED) Area 10340, Internal Entry Descriptor (IED) Area 10342, S-Op Code Area 10344, Procedure Environment Descriptor (PED) 10348, Name Table (NT) 10350, and Access Information Array (AIA) Area 10352.

Header 10338 contains certain information identifying PO 10318 and indicating the number of entries in EED area 10340, discussed momentarily.

60 EED area 10340 and IED area 10342 together contain an Entry Descriptor (ED) for each procedure in KOSPO 10318. KOSPO 10318 is represented as containing Procedures 1, 2, and 11, of which Procedure 11 will be used as an example in the present discussion. EDs effectively comprise an index through certain all information in KOSPO 10318 can be located. IEDs form an index to all KOSPO 10318 procedures which may be called only from other procedures contained in KOSPO 10318. EEDs form an index to all KOSPO 10318 procedures which may be called by procedures external to KOSPO 10318. Externally callable procedures are distinguished and, as described in a following discussion of CS 10110's protection mechanisms, in 65

confirming external calling procedure's access rights.

Referring to ED 11, ED for procedure 11, three fields are shown therein. Procedure Environment Descriptor Offset (PEDO) field indicates the start, relative to start of KOSPO 10318, of Procedure 11's PED in PED Area 10348. As will be discussed further below, a procedure's PED contains a set of pointers for locating information used in the execution of that procedure. PED Area 10348 contains a PED for each procedure contained in 10318. In the present embodiment of CS 10110, a single PED may be shared by two or more procedures. Code Entry Point (CEP) field indicates the start, relative to Procedure Base Pointer (PBP) which will be discussed below, of Procedure 11's SIN Code and SIN Code Area 10344. Finally, ED 11's Initial Frame Size (IFS) field indicates the required Initial Frame Size of the KOSMAS 10334 frame storing Procedure 11's automatic data.

PED 11, Procedure 11's PED in PED Area 10348, contains a set of pointers for locating information used in execution of Procedure 11. The first entry in PED 11 is a header containing information identifying PED 11. PED 11's Procedure Base Pointer (PBP) entry is a pointer providing a fixed reference from which other information in PO 10318 may be located. In a specific example, Procedure 11's CEP indicates the location, relative to PBP, of the start of Procedure 11's S-Op code in S-Op Code Area 10344. As will be described further below, PBP is a CS 10110 Architectural Base Pointer (ABP). CS 10110's ABP's are a set of architectural pointers used in CS 10110 to facilitate addressing of CS 10110's address space. PED 11's Static Data Pointer (SDP) entry points to data, in PO 10318, specifying certain parameters of P 10310's KOSSDA 10326. Name Table Pointer (NTP) entry is a pointer indicating the location, in NT 10350, of Name Table Entry's (NTE's) for Procedure 11's operands. NT 10350 and NTE's will be described in greater detail in the following discussion of Computer System 10110's Addressing Structure. PED 11's S-Interpreter Pointer (SIP) entry is a pointer, discussed in greater detail in a following discussion of CS 10110's microcode structure, pointing to the particular S-Interpreter (SINT) to be used in interpreting Procedure 11's SIN Code.

Referring finally to AIA 10352, AIA 10352 contains, as previously discussed, information pertaining to access rights required of any external procedure calling a 10318 procedure. There is an AIA 10352 entry for each PO 10318 procedure which may be called by an external procedure. A particular AIA entry may be shared by one or more procedures having an ED in EED Area 10340. Each EED contains certain information, not shown for clarity of presentation, indicating that that procedure's corresponding AIA entry must be referred to, and the calling procedure's access rights confirmed, whenever that procedure is called.

2. Stack Mechanism (Figs. 104, 105)

As previously described, P10310's stack mechanisms include SS 10336, used in part for storing machine state, and MAS's 10328 to 10334, used to store local data generated during execution of P 10310's procedures. P 10310 is represented as containing an MAS for each CS 10110 domain. In an alternate embodiment of CS 10110, a particular P 10310 will include MAS's only for those domains in which that P 10310 is executing a procedure.

Referring to MAS's 10328 to 10334 and SS 10336, P 10310 is represented as having had eleven procedure calls. Procedure 0 has called Procedure 1, Procedure 1 has called Procedure 2, and so on. Each time a procedure is called, a corresponding stack frame is constructed on the MAS of the domain in which the called procedure is executed. For example, Procedures 1, 2, and 11 execute in KOS domain; MAS frames for Procedures 1, 2, and 11 therefore are placed on KOSMAS 10334. Similarly, Procedures 3 and 9 execute in EOS domain, so that their stack frames are placed on EOSMAS 10332. Procedures 5 and 6 execute in DBMS domain, so that their stack frames are placed on DBMSMAS 10330. Procedures 4, 7, 8, and 10 execute in User domain with their stack frames being placed on USERMAS 10328. Procedure 11 is the most recently called procedure and procedure 11's stack frame on KOSMAS 10334 is referred to as the current frame. Procedure 11 is the procedure which is currently being executed when VP 10310 is bound to CS 10110.

SS 10336, which is a stack mechanism of CS 10110 micromachine, contains a frame for each of Procedures 1 to 11. Each SS 10336 frame contains, in part, CS 10110 operating state for its corresponding procedure.

Referring to Fig. 104, a schematic representation of a typical MAS, for example KOSMAS 10334, is shown. KOSMAS 10334 includes Stack Header 10410 and a Frame 10412 for each procedure on KOSMAS 10334. Each Frame 10412 includes a Frame Header 10414, and may contain a Linkage Pointer Block 10416, a Local Pointer Block 10418, and a Local (Automatic) Data Block 10420.

KOSMAS 10334 Stack Header 10410 contains at least the following information:

- (1) an offset, relative to Stack Header 10410, indicating the location of Frame Header 10414 of the first frame on KOSMAS 10334;
- (2) a Stack Top Offset (STO) indicating location, relative to start of KOSMAS 10334, of the top of KOSMAS 10334; top of KOSMAS 10334 is indicated by pointer STO pointing to the top of the last entry of Procedure 11 Frame 10412's Local Data Block 10420;
- (3) an offset, relative to start of KOSMAS 10334, indicating location of Frame Header 10414 of the current top frame of KOSMAS 10334; in Fig. 104 this offset is represented by Frame Pointer (FP), an ABP discussed further below;
- (4) the VP 10310's UID;
- (5) a UID Pointer indicating location of certain domain environment information, described further in a

following discussion;

(6) a signaller pointer indicating the location of certain routines for handling certain CS 10110 operating system faults;

(7) a UID pointer indicating location of KOSSDA 10326; and

5 (8) a frame label sequencer containing pointers to headers of frames in other domains; these pointers are used in executing non-local go-to operations.

KOSMAS 10334 Stack Header 10410 thereby contains information for locating certain important points in KOSMAS 10334's structure, and for locating certain information pertinent to executing procedures in KOS domain.

10 Each Frame Header 10414 contains at least the following information:

(1) offsets, relative to the Frame Header 10414, indicating the locations of Frame Headers 10414 of the previous and next frames of KOSMAS 10334;

(2) an offset, relative to the Frame Header 10414, indicating the location of the top of that Frame 10412;

(3) information indicating the number of passed arguments contained in that Frame 10412;

15 (4) a dynamic back pointer, in UID/Offset format, to the previous Frame 10412 if that previous Frame 10412 resides in another domain;

(5) a UID/Offset pointer to the environmental descriptor of the procedure calling that procedure;

20 (6) a frame label sequence containing information indicating the locations of other Frame Headers 10414 in KOSMAS 10334; this information is used to locate other frames in KOSMAS 10334 for the purpose of executing local go-to operations. Frame Headers 10414 thereby contain information for locating certain important points in KOSMAS 10334 structure, and certain data pertinent to executing the associated procedures. In addition, Frame Headers 10414, in combination with Stack Header 10410, contain information for linking the activation records of each VP 10310 MAS, and for linking together the activation records of the individual MAS's.

25 Linkage Pointer Blocks 10416 contain pointers to arguments passed from a calling procedure to the called procedure. For example, Linkage Pointer Block 10416 of Procedure 11's Frame 10412 will contain pointers to arguments passed to Procedure 11 from Procedure 10. The use of linkage pointers in CS 10110's addressing structure will be discussed further in a following discussion of CS 10110's Addressing Structure. Local Data Pointer Blocks 10418 contain pointers to certain of the associated procedure's local data. Indicated in Fig. 104 is a pointer, Frame Pointer (FP), pointing between top most Frame 10412's Linkage Pointer Block 10416 and Local Data Pointer Block 10418. FP, described further in following discussions, is an ABP to MAS Frame 10412 of the process's current procedure.

Each Frame 10412's Local (Automatic) Data Block 10420 contains certain of the associated procedure's automatic data.

35 As described above, at each procedure call a MAS frame is constructed on top of the MAS of the domain in which the called procedure is executed. For example, when Procedure 10 calls Procedure 11 a Frame Header 10414 for Procedure 11 is constructed and placed on KOSMAS 10334. Procedure 11's linkage pointers are then generated, and placed in Procedure 11's Linkage Pointer Block 10416. Next Procedure 11's local pointers are generated and placed in Procedure 11's Local Pointer Block 10418. Finally, Procedure 11's local data is placed in Procedure 11's Local Data Block 10420. During this operation, USERMAS 10328's frame label sequence is updated to include an entry pointing to Procedure 11's Frame Header 10414. KOSMAS 10334's Stack Header 10410 is updated with respect to STO to the new top of KOSMAS 10334. Procedure 2's Frame Header 10414 is updated with respect to offset to Frame Header 10414 of Procedure 11 Frame 10412, and with respect to frame label sequence indicating location of Procedure 11's Frame Header 45 10414. As Procedure 11 is then the current procedure, FP is updated to a point between Linkage Pointer Block 10416 and Local Pointer Block 10418 of Procedure 11's Frame 10412. Also, as will be discussed below, a new frame is constructed on SS 10336 or Procedure 11. CS 10110 will then proceed to execute Procedure 11. During execution of Procedure 11, any further local data generated may be placed on the top of Procedure 11's Local Data Block 10420. The top of stack offset information in Procedure 11's Frame Header 50 10414 and in KOSMAS 10334 Stack Header 10410 will be updated accordingly.

MAS's 10328 to 10334 thereby provide a per domain stack mechanism for storing data pertaining to individual procedures, thus allowing stacking of procedures without loss of this data. Although structured on a domain basis, MAS's 10328 to 10334 comprise a unified logical stack structure threaded together through information stored in MAS stack and frame headers.

55 As described above and previously, SS 10336 is a CS 10110 micromachine stack structure for storing, in part, CS 10110 micromachine state for each stacked VP 10310 procedure. Referring to Fig. 105, a partial schematic representation of a SS 10336 Stack Frame 10510 is shown. SS 10336 Stack Header 10512 and Frame Headers 10514 contain information similar to that in MAS Stack Headers 10410 and Frame Headers 10414. Again, the information contained therein locates certain points within SS 10336 structure, and threads together SS 10336 with MAS's 10328 to 10334.

60 SS 10336 Stack Frame 10510 contains certain information used by the CS 10110 micromachine in executing the VP 10212 procedure with which this frame is associated. Procedure Pointer Block 10516 contains certain pointers including ABPs, used by CS 10110 micromachine in locating information within VP 10310's information structures. Micro-Routine Frames (MRFs) 10518 together comprise Micro-Routine Stack (MRS) 10520 within each SS 10336 Stack Frame 10510. MRS Stack 10520 is associated with the 65

internal operation of CS 10110 microroutines executed during execution of the VP 10212 procedure associated with the Stack Frame 10510. SS 10336 is thus a dual function CS 10110 micromachine stack. Pointer Block 10516 entries effectively define an interface between CS 10110 micromachine and the current procedure of the current process. MRS 10520 comprise a stack mechanism for the internal operations of CS 10110 micromachine.

Having briefly described Virtual Processes 10212, FURSM 10214 will be described next. As stated above, EURSM 10216 is similar in operation to FURSM 10214 and will be described in following detailed descriptions of CS 10110 structure and operation.

3. FURSM 10214 (Fig. 103)

Referring again to Fig. 103, FURSM 10214 includes CS 10110 micromachine information structures used internally to CS 10110 micromachine in executing the procedures of a P 10310. When a VP, for example P 10310, is to be executed, certain information regarding that VP is transferred from the Virtual Processes 10212 to FURSM 10214 for use in executing that procedure. In this respect, FURSM 10214 may be regarded as an acceleration mechanism for the current Virtual Process 10212.

FURSM 10214 includes General Register File (GRF) 10354, Micro Stack Pointer Register Mechanism (MISPR) 10356, and Return Control Word Stack (RCWS) 10358. GRF 10354 includes Global Registers (GRs) 10360 and Stack Registers (SRs) 10362. GRs 10360 include Architectural Base Registers (ABRs) 10364 and Micro-Control Registers (MCRs) 10366. Stack Registers 10362 include Micro-Stack (MIS) 10368 and Monitor Stack (MOS) 10370.

Referring first to GRF 10354, and assuming for example that Procedure 11 of P 10310 is currently being executed, GRF 10354 primarily contains certain pointers to P 10310 data used in execution of Procedure 11. As previously discussed, CS 10110's addressing structure includes certain Architectural Base Pointers (ABPs) for each procedure. ABPs provide a framework for accessing CS 10110's address space. The ABPs of each procedure include a Frame Pointer (FP), a Procedure Base Pointer (PBP), and a Static Data Pointer (STP). As discussed above with reference to KOSPO 10318, these ABPs reside in the procedure's PEDs. When a procedure is called, these ABPs are transferred from that procedure's PED to ABR's 10364 and reside therein for the duration of that procedure. As indicated in Fig. 103, FP points between Linkage Pointer Block 10416 and Local pointer Blocks 10418 of Procedure 11's Frame 10412 on KOSMAS 10334. PBP points to the reference point from which the elements of KOSPO 10318 are located. SDP points to KOSSDA 10326. If Procedure 11 calls, for example, a Procedure 12, Procedure 11's ABPs will be transferred onto Procedure Pointer Block 10516 of SS 10336 Stack Frame 10510 for Procedure 11. Upon return to Procedure 11, Procedure 11's ABPs will be transferred from Procedure Pointer Block 10516 to ABR's 10364 and execution of Procedure 11 resumed.

MCRs 10366 contain certain pointers used by CS 10110 micromachine in executing Procedure 11. CS 10110 micromachine pointers indicated in Fig. 103 include Program Counter (PC), Name Table Pointer (NTP), S-Interpreter Pointer (SIP), Secure Stack Pointer (SSP), and Secure Stack Top Offset (SSTO). NTP and SIP have been previously described with reference to KOSPO 10318 and reside in KOSPO 10318. NTP and SIP are transferred into MCR's 10366 at start of execution of Procedure 11. PC, as indicated in Fig. 103, is a pointer to the Procedure 11 SIN currently being executed by CS 10110. PC is initially generated from Procedure 11's PBP and CEP and is thereafter incremented by CS 10110 micromachine as Procedure 11's SIN sequences are executed. SSP and SSTO are, as described in a following discussion, generated from information contained in SS 10336's Stack Header 10512 and Frame Headers 10514. As indicated in Fig. 103 SSP points to start of SS 10336 while SSTO indicates the current top frame on SS 10336, whether Procedure Pointer Block 10516 or a MRF 10518 of MRS 10520, by indicating an offset relative to SSP. If Procedure 11 calls a subsequent procedure, the contents of MCR's 10366 are transferred into Procedure 11's Procedure Pointer Block 10516 on SS 10336, and are returned to MCR's 10366 upon return to Procedure 11.

Registers 10360 contain further pointers, described in following detailed discussions of CS 10110 operation, and certain registers which may be used to contain the current procedure's local data.

Referring now to Stack Registers 10362, MIS 10368 is an upward extension, or acceleration, of MRS 10520 of the current procedure. As previously stated, MRS 10520 is used by CS 10110 micromachine in executing certain microroutines during execution of a particular procedure. MIS 10368 enhances the efficiency of CS 10110 micromachine in executing these microroutines by accelerating certain most recent MRFs 10518 of that procedure's MRS 10520 into FU 10120. MIS 10368 may contain, for example, up to the eight most recent MRFs 10518 of the current procedures MRS 10520. As various microroutines are called or returned from, MRS 10520 MRF's 10518 are transferred accordingly between SS 10336 and MIS 10368 so that MIS 10368 always contains at least the top MRF 10518 of MRS 10520, and at most eight MRFs 10518 of MRS 10520. MISPR 10356 is a CS 10110 micromachine mechanism for maintaining MIS 10368. MISPR 10356 contains a Current Pointer, a Previous Pointer, and a Bottom Pointer. Current Pointer points to the top-most MRF 10518 on MIS 10368. Previous Pointer points to the previous MRF 10518 on MIS 10368, and Bottom Pointer points to the bottom-most MRF 10518 on MIS 10368. MISPR 10356's Current, Previous and Bottom Pointers are updated as MRFs 10518 are transferred between SS 10336 and MIS 10368. If Procedure 11 calls a subsequent procedure, all Procedure 11 MRFs 10518 are transferred from MIS 10368 to Procedure 11's MRS 10520 on SS 10336. Upon return to Procedure 11, up to seven of Procedure 11's MRFs 10518

frames are returned from SS 10336 to MIS 10368.

Referring to MOS 10370, MOS 10370 is a stack mechanism used by CS 10110 micromachine for certain microroutines for handling fault or error conditions. These microroutines always run to completion, so that MOS 10370 resides entirely in FM 10120 and is not an extension of a stack residing in a P 10310 in MEM 10112. MOS 10370 may contain, for example, eight frames. If more than eight successive fault or error conditions occur, this is regarded as a major failure of CS 10110. Control of CS 10110 may then be transferred to DP 10118. As will be described in a following discussion, diagnostic programs in DP 10118 may then be used to diagnose and locate the CS 10110 faults or errors. In other embodiments of CS 10110 MOS 10370 may contain more or fewer stack frames, depending upon the degree of self diagnosis and correction capability desired for CS 10110.

RCWS 10358 is a two-part stack mechanism. A first part operates in parallel with MIS 10368 and a second part operates in parallel with MOS 10370. As previously described, CS 10110 is a microcode controlled system. RCWS is a stack for storing the current microinstruction being executed by CS 10110 micromachine when the current procedure is interrupted by a fault or error condition, or when a subsequent procedure is called. That portion of RCWS 10358 associated with MIS 10368 contains an entry for each MRF 10518 residing in MIS 10368. These RCWS 10358 entries are transferred between SS 10336 and MIS 10368 in parallel with their associated MRFs 10518. When resident in SS 10336, these RCWS 10358 entries are stored within their associated MRFs 10518. That portion of RCWS 10358 associated with MOS 10370 similarly operates in parallel with MOS 10370 and, like MOS 10370, is not an extension of an MEM 10112 resident stack.

In summary, each process active in CS 10110 exists as a separate, complete, and self-contained entity, or Virtual Process, and is structurally organized on a domain basis. Each Virtual Process includes, besides procedure and data objects, a set of MAS's for storing local data of that processes procedures. Each Virtual Process also includes a CS 10110 micromachine stack, SS 10336, for storing CS 10110 micromachine state pertaining to each stacked procedure of the Virtual Process. CS 10110 micromachine includes a set of information structures, register 10360, MIS 10368, MOS 10370, and RCWS 10358, used by CS 10110 micromachine in executing the Virtual Process's procedures. Certain of these CS 10110 micromachine information structures are shared with the currently executing Virtual Process, and thus are effectively acceleration mechanisms for the current Virtual Process, while others are completely internal to CS 10110 micromachine.

A primary feature of CS 10110 is that each process' macrostacks and secure stack resides in MEM 10112. CS 10110's macrostack and secure stacks are therefore effectively unlimited in depth.

Yet another feature of CS 10110 micromachine is the use of GRF 10354. GRF 10354 is, in an embodiment of CS 10110, a unitary register array containing for example, 256 registers. Certain portions, or address locations, of GRF 10354 are dedicated to, respectively, GRs 10360, MIS 10368, and MOS 10370. The capacities of GR 10360, MIS 10368, and MOS 10370, may therefore be adjusted, as required for optimum CS 10110 efficiency, by reassignment of GRF 10354's address space. In other embodiments of CS 10110, GRs 10360, MIS 10368, and MOS 10370 may be implemented a functionally separate registers arrays.

Having briefly described the structure and operation of Process Structures 10210, VP State Block 10218 will be described next below.

C. Virtual Processor State Blocks and Virtual Process Creation (Fig. 102)

Referring again to Fig. 102, VP State Blocks 10218 is used in management and control of processes. VP State Blocks 10218 contains a VP State Block for each Virtual Process (VP) selected for execution by CS 10110. Each such VP State Block contains at least the following information: (1) the state, or identification number of a VP;

- (2) entries identifying the particular principle and particular process of the VP;
- (3) an AON pointer to that VP's secure stack (e.g., SS 10336);
- (4) the AON's of that VP's MAS stack objects (e.g., MAS's 10328 to 10334); and,
- (5) certain information used by CS 10110's VP Management System.

The information contained in each VP State Block thereby defines the current state of the associated VP.

A Process is loaded into CS 10110 by building a primitive access record and loading this access record into CS 10110 to appear as an already existing VP. A VP is created by creating a Process Object, including pointers to macro- and secure-stack objects created for that VP, micromachine state entries, and a pointer to the user's program. CS 10110's KOS then generates Macro- and Secure-Stack Objects with headers for that process and, as described further below, loads protection information regarding that process' objects into protection Structures 10230. CS 10110's KOS then copies this primitive machine state record into a vacant VPSB selected by CS 10110's VP Manager, thus binding the newly created VP into CS 10110. At that time a KOS Initializer procedure completes creation of the VP for example by calling in the user's program through a compiler. The newly created VP may then be executed by CS 10110.

Having briefly described VP State Blocks 10218 and creation of a VP, CS 10110's Addressing Structures 10220 will be described next below.

D. Addressing Structure 10220 (Figs. 103, 106, 107, 108)

1. Objects, UID's, AON's, Names, and Physical Addresses (Fig. 106)

As previously described, the data space accessible to CS 10110 is divided into segments, or containers, referred to as objects. In an embodiment of CS 10110, the addressable data space of each object has a capacity of 2^{32} bits of information and is structured into 2^{18} pages with each page containing 2^{14} bits of information.

Referring to Fig. 106A, a schematic representation of CS 10110's addressing structure is shown. Each object created for use in, or by operation of, a CS 10110 is permanently assigned a unique identifier (UID). An object's UID allows an object to be uniquely identified and located at any future point in time. Each UID is an 80 bit number, so that the total addressable space of all CS 10110's includes 2^{80} objects wherein each object may contain up to 2^{32} bits of information. As indicated in Fig. 106, each 80 bit UID is comprised of 32 bits of Logical Allocation Unit Identifier (LAUID) and 48 bits of Object Serial Number (OSN). LAUIDs are associated with individual CS 10110 systems. LAUIDs identify the particular CS 10110 system generating a particular object. Each LAUID is comprised of a Logical Allocation Unit Group Number (LAUGN) and a Logical Allocation Unit Serial Number (LAUSN). LAUGNs are assigned to individual CS 10110 systems and may be guaranteed to be unique to a particular system. A particular system may, however, be assigned more than one LAUGN so that there may be a time varying mapping between LAUGNs and CS 10110 systems. LAUSNs are assigned within a particular system and, while LAUSNs may be unique within a particular system, LAUSNs need not be unique between systems and need not map onto the physical structure of a particular system.

OSNs are associated with individual objects created by an LAU and are generated by an Architectural Clock in each CS 10110. Architectural clock is defined as a 64 bit binary number representing increasing time. Least significant bit of architectural clock represents increments of 600 picoseconds, and most significant bit represents increments of 127 years. In the present embodiment of CS 10110, certain most significant and least significant bits of architectural clock time are disregarded as generally not required practice. Time indicated by architectural clock is measured relative to an arbitrary, fixed point in time. This point in time is the same for all CS 10110s which will ever be constructed. All CS 10110s in existence will therefore indicate the same architectural clock time and all UIDs generated will have a common basis. The use of an architectural clock for generation of OSNs is advantageous in that it avoids the possibility of accidental duplication of OSNs if a CS 10110 fails and is subsequently reinitiated.

As stated above, each object generated by or for use in a CS 10110 is uniquely identified by its associated UID. By appending Offset (O) and Length (L) information to an object's UID, a UID logical address is generated which may be used to locate particular segments of data residing in a particular object. As indicated in Fig. 106, O and L fields of a UID logical address are each 32 bits. O and L fields can therefore indicate any particular bit, out of 2^{32-1} bits, in an object and thus allow bit granular addressing of information in objects.

As indicated in Fig. 106 and as previously described, each object active in CS 10110 is assigned a short temporary unique identifier valid only within JP 10114 and referred to as an Active Object Number (AON). Because fewer objects may be active in a CS 10110 than may exist in a CS 10110's address space, AON's are, in the present embodiment of CS 10110, 14 bits in length. A particular CS 10110 may therefore contain up to 2^{14} active objects. An object's AON is used within JP 10114 in place of that object's UID. For example, as discussed above with reference to process structures 10210, a procedure's FP points to start of that procedure's frame on its process' MAS. When that FP is residing in SS 10336, it is expressed as a UID. When that procedure is to be executed, FP is transferred from SS 10336 to ABR's 10364 and is translated into the corresponding AON. Similarly, when that procedure is stacked, FP is returned to SS 10336 and in doing so is translated into the corresponding UID. Again, a particular data segment in an object may be addressed by means of an AON logical address comprising the object's AON plus associated 32 bit Offset (O) and Length (L) fields.

Each operand appearing in a process is assigned a Name and all references to a process's operands are through those assigned Names. As indicated in Fig. 106B, in the present embodiment of CS 10110 each Name is an 8, 12, or 16 bit number. All Names within a particular process will be of the same length. As will be described in a following discussion, Names appearing during execution of a process may be resolved, through a procedure's Name Table 10350 or through Name Cache 10226, to an AON logical address. As described below, an AON logical address corresponding to an operand Name may then be evaluated to a MEM 10112 physical address to locate the operand referred to.

The evaluation of AON logical addresses to MEM 10112 physical addresses is represented in Fig. 106C. An AON logical address's L field is not involved in evaluation of an AON logical address to a physical address and, for purposes of clarity of presentation, is therefore not represented in Fig. 106C. AON logical address L field is to be understood to be appended to the addresses represented in the various steps of the evaluation procedure shown in Fig. 106C.

As described above, objects are 2^{32} bits structured into 2^{18} pages with each page containing a 2^{14} bits of data. MEM 10112 is similarly physically structured into frames with, in the present embodiment of CS 10110, each frame containing 2^{14} bits of data. In other embodiments of CS 10110, both pages and frames may be of different sizes but the translation of AON logical addresses to MEM 10112 physical addresses will be similar to that described momentarily.

An AON logical address O field was previously described as a 32 bit number representing the start, relative to start of the object, of the addressed data segment within the object. The 18 most significant bits of O field represent the number (P) of the page within the object upon which the first bit of the addressed data occurs. The 14 least significant bits of O field represent the offset (O_p), relative to the start of the page, within that page of the first bit of the addressed data. AON logical address O field may therefore, as indicated in Fig. 106C, be divided into an 18 bit page (P) field and a 14 bit offset within page (O_p) field. Since, as described above, MEM 10112 physical frame size is equal to object page size, AON logical address O_p field may be used directly as an offset within frame (O_f) field of the physical address. As will be described below, an AON logical address AON and P fields may then be related to the frame number (FN) of the MEM 10112 frame in which that page resides, through Addressing Mechanisms 10220.

Having briefly described the relationships between UIDs, UID Logical Addresses, Names, AONs, AON Logical Addresses, and MEM 10112 Physical Addresses, Addressing Mechanisms 10220 will be described next below.

2. Addressing Mechanisms 10220 (Fig. 107)

Referring to Fig. 107, a schematic representation of Computer System 10110's Addressing Mechanisms 10220 is shown. As previously described, Addressing Mechanisms 10220 comprise UID/AON Tables 10222, Memory Management Tables 10224, Name Cache 10226, and Address Translation Unit 10228.

UID/AON Tables 10222 relate each object's UID to its assigned AON and include AOT Hash Table (AOTHT) 10710, Active Object Table (AOT) 10712, and Active Object Table Annex (AOTA) 10714.

An AON corresponding to a particular UID is determined through AOTHT 10710. The UID is hashed to provide a UID index into AOTHT 10710, which then provides the corresponding AON. AOTHT 10710 is effectively an acceleration mechanism of AOT 10712 to, as just described, provide rapid translation of UIDs to AONs. AONs are used as indexes into AOT 10712, which provides a corresponding AOT Entry (AOTE). An AOTE as described in following detailed discussions of CS 10110, includes, among other information, the UID corresponding to the AON indexing the AOTE. In addition to providing translation between AONs and UIDs, the UID of an AOTE may be compared to an original UID to determine the correctness of an AON from AOTHT 10710.

Associated with AOT 10712 is AOTA 10714. AOTA 10714 is an extension of AOT 10712 and contains certain information pertaining to active objects, for example the domain of execution of each active procedure object.

Having briefly described CS 10110's mechanism for relating UIDs and AONs, CS 10110's mechanism for resolving operand Names to AON logical addresses will be described next below.

3. Name Resolution (Figs. 103, 108)

Referring first to Fig. 103, each procedure object in a VP, for example KOSPO 10318 in VP 10310, was described as containing a Name Table (NT) 10350. Each NT 10350 contains a Name Table Entry (NTE) for each operand whose Name appears its procedure. Each NTE contains a description of how to resolve the corresponding Name to an AON Logical Address, including fetch mode information, type of data referred to by that Name, and length of the data segment referred to.

Referring to Fig. 108, a representation of an NTE is shown. As indicated, this NTE contains seven information fields: Flag, Base (B), Predisplacement (PR), Length (L), Displacement (D), Index (I), and Inter-element Spacing (IES). Flag Field, in part, contains information describing how the remaining fields of the NTE are to be interpreted, type of information referred to by the NTE, and how that information is to be handled when fetched from MEM 10112. L Field, as previously described, indicates length, or number of bits in, the data segment. Functions of the other NTE fields will be described during the following discussions.

In a present embodiment of CS 10110, there are five types of NTE: (1) base (B) is not a Name, address resolution is not indirect; (2) B is not a Name, address resolution is indirect; (3) B is a Name, address resolution is indirect; (4) B is a Name, address resolution is indirect. A fifth type is an NTE selecting a particular element from an array of elements. These five types of NTE and their resolution will be described below, in the order mentioned.

In the first type, B is not a Name and address resolution is not indirect, B Field specifies an ABR 10364 containing an AON plus offset (AON/O) Pointer. The contents of D Field are added to the O Field of this pointer, and the result is the AON Logical Address of the operand. In the second type, B is not a Name and address resolution is indirect, B Field again specifies an ABR 10364 containing an AON/O pointer. The contents of PR Field are added to the O Field of the AON/O pointer to provide an AON Logical Address of a Base Pointer. The Base Pointer AON Logical Address is evaluated, as described below, and the Base Pointer fetched from MEM 10112. The contents of D Field are added to the O Field of the Base Pointer and the result is the AON Logical Address of the operand.

NTE types 3 and 4 correspond, respectively to NTE types 1 and 2 and are resolved in the same manner except that B Field contains a Name. The B Field Name is resolved through another NTE to obtain an AON/O pointer which is used in place of the ABR 10364 pointers referred to in discussion of types 1 and 2.

The fifth type of NTE is used in references to elements of an array. These array NTEs are resolved in the

same manner as NTE types 1 through 4 above to provide an AON Logical Address of the start of the array. I and IES Fields provide additional information to locate a particular element in the array. I Field is always Name which is resolved to obtain an operand value representing the particular element in the array. IES Field provides information regarding spacing between elements of the array, that is the number of bits between adjacent element of the array. IES Field may contain the actual IES value, or it may contain a Name which is resolved to an AON Logical Address leading to the inter-element spacing value. The I and IES values, obtained by resolving the I and IES Fields as just described, are multiplied together to determine the offset, relative to the start of the array, of the particular element referred to by the NTE. This within array offset is added to the O Field of the AON Logical Address of the start of the array to provide the AON Logical Address of the element.

In the current embodiment of CS 10110, certain NTE fields, for example B, D, and Flag fields, always contain literals. Certain other fields, for example, IES, D, PRE, and L fields, may contain either literals or names to be resolved. Yet other fields, for example I field, always contain names which must be resolved.

Passing of arguments from a calling procedure to a called procedure has been previously discussed with reference to Virtual Processes 10212 above, and more specifically with regard to MAS's 10328 to 10334 of VP 10310. Passing of arguments is accomplished through the calling and called procedure's Name Tables 10350. In illustration, a procedure W(a, b, c) may wish to pass arguments a, b, and c to procedure X(u, v, w), where arguments, v and w correspond to arguments a, b, and c. At compilation, NTEs are generated for arguments a, b, and c in procedure W's procedure object, and NTEs are generated for arguments u, v and w in Procedure X's procedure object. Procedure X's NTEs for u, v, and w are constructed to resolve to point to pointers in Linkage Pointer Block 10416 of Procedure X's Frame 10412 in MAS. To pass arguments a, b, and c from Procedure W to Procedure X, the NTEs of arguments a, b, and c are resolved to AON Logical Addresses (i.e., AON/O form). Arguments a, b, and c's AON Logical Addresses are then translated to corresponding UID addresses which are placed in Procedure X's Linkage Pointer Block 10416 at those places pointed to by Procedure X's NTEs for u, v, and w. When Procedure X is executed, the resolution of Procedure X's NTEs for u, v, and w will be resolved to locate the pointers, in Procedure X's Linkage Pointer Block 10416 to arguments a, b, and c. When arguments are passed in this manner, the data type and length information are obtained from the called procedure's NTEs, rather than the calling procedure's NTEs. This allows the calling procedure to pass only a portion of, for example, arguments a, b, or c, to the called procedure and thus may be regarded as a feature of CS 10110's protection mechanisms.

Having briefly described resolution of Names to AON/Offset addresses, and having previously described translation of UID addresses to AON addresses, the evaluation of AON addresses to MEM 10112 physical addresses will be described next below.

4. Evaluation of AON Addresses to Physical Addresses (Fig. 107)

Referring again to Fig. 107, a partial schematic representation of CS 10110's Memory Management Table 10224 is shown. Memory Hash Table (MHT) 10716 and Memory Frame Table (MFT) 10718 are concerned with translation of AON addresses into MEM 10112 physical addresses and will be discussed first. Working Set Matrix (WSM) 10720 and Virtual Memory Manager Request Queue (VMMRQ) 10722 are concerned with management of MEM 10112's available physical address base and will be discussed second. Active Object Request Queue (AORQ) 10728 and Logical Allocation Unit Directory (LAUD) 10730 are concerned with locating inactive objects and management of which objects are active in CS 10110 and will be discussed last.

Translation of AON/O Logical Addresses to MEM 10112 physical addresses was previously discussed with reference to Fig. 106C. As stated in that discussion, objects are divided into pages. Correspondingly, the AON/O Logical Address' O Field is divided into an 18 bit page number (P) Field and a 14 bit offset within a page (O_p) Field. MEM 10112 is structured into frames, each of which in the present embodiment of CS 10110 is equal to a page of an object. An AON/O address' O_p Field may therefore be used directly as an offset within frame (O_f) of the corresponding physical address. The AON and P fields of an AON address must, however, be translated into a MEM 10112 frame represented by a corresponding Frame Number (FN).

Referring now to Fig. 107, an AON address' AON and P Fields are "hashed" to generate an MHT index which is used as an index into MHT 10716. Briefly, "hashing" is a method of indexing, or locating, information in a table wherein indexes to the information are generated from the information itself through a "hashing function". A hashing function maps each piece of information to the corresponding index generated from it through the hashing function. MHT 10716 then provides the corresponding FN of the MEM 10112 frame in which that page is stored. FNs are used as indexes into MFT 10718, which contains, for each FN, an entry describing the page stored in that frame. This information includes the AON and P of the page stored in that MEM 10112 frame. An FN from MHT 10716 may therefore be used as an index into MFT 10718 and the resulting AON/P of MFT 10718 compared to the original AON/P to confirm the correctness of the FN obtained from MHT 10716. MHT 10716 is an effectively acceleration mechanism of MFT 10718 to provide rapid translation of AON address to MEM 10112 physical addresses.

MFT 10718 also stores "used" and "modified" information for each page in MEM 10112. This information indicates which page frames stored therein have been used and which have been modified.

This information is used by CS 10110 in determining which frames may be deleted from MEM 10112, or are free, when pages are to be written into MEM 10112 from backing store (ED 10124). For example, if a page's modified bit indicates that that page has not been written into, it is not necessary to write that page back into backing store when it is deleted from MEM 10112; instead, that page may be simply erased.

5 Referring finally to ATU 10228, ATU 10228 is an acceleration mechanism for MHT 10716. AON/O addresses are used directly, without hashing, as indexes into ATU 10228 and ATU 10228 correctly provides corresponding FN and O outputs. A CS 10110 mechanism, described in a following detailed discussion of CS 10110 operation, continually updates the contents of ATU 10228 so that ATU 10228 contain the FN's and O_p (O_p) of the pages most frequently referenced by the current process. If ATU 10228 does not contain a
10 corresponding entry for a given AON input, an ATU fault occurs and the FN and O information may be obtained directly from MHT 10716.

Referring now to WSM 10720 and VMMRQ 10722, as previously stated these mechanisms are concerned with the management of MEM 10112's available address space. For example, if MHT 10716 and MFT 10718 do not contain an entry for a page referenced by the current procedure, an MHT/MFT fault
15 occurs and the reference page must be fetched from backing store (ED 10124) and read into MEM 10112. WSM 10720 contains an entry for each page resident in MEM 10112. These entries are accessed by indexes comprising the Virtual Processor Number (VPN) of the virtual process making a page reference and the P of the page being referenced. Each WSM 10720 entry contains 2 bits stating whether the particular page is part of a VP's working set, that is, used by that VP, and whether that page has been referenced by that VP.
20 This information, together with the information contained in that MFT 10718 entries described above, is used by CS 10110's Virtual Memory Manager (VMM) in transferring pages into and out of MEM 10112.

CS 10110's VMM maintains VMMRQ 10722, which is used by VMM to control transfer of pages into and out of MEM 10112. VMMRQ 10722 includes Virtual Memory Request Counter (VMRC) 10724 and a Queue of Virtual Memory Request Entries (VMREs) 10726. As will be discussed momentarily, VMRC 10724 tracks the
25 number of currently outstanding request for pages. Each VMRE 10726 describes a particular page which has been requested. Upon occurrence of a MHT/MFT (or page) fault, VMRC 10724 is incremented, which initiates operation of CS 10110's VMM, and a VMRE 10726 is placed in the queue. Each VMRE 10726 comprises the VPN of the process requesting the page and the AON/O of the page requested. At this time, the VP making the request is swapped out of JP 10114 and another VP bound to JP 10114. VMM allocates
30 MEM 10112 frame to contain the requested page, using the previously described information in MFT 10718 and WSM 10720 to select this frame. In doing so, VMM may discard a page currently resident in MEM 10112 for example, on the basis of being the oldest page, an unused page, or an unmodified page which does not have to be written back into backing store. VMM then requests an I/O operation to transfer the requested page into the frame selected by the VMM. While the I/O operation is proceeding, VMM generates new entries in MHT 10716 and MFT 10718 for the requested page, cleans the frame in MEM 10112 which is to be
35 occupied by that page, and suspends operation. IOS 10116 will proceed to execute the I/O operation and writes the requested page directly into MEM 10112 in the frame specified by VMM. IOS 10116 then notifies CS 10110's VMM that the page now resides in memory and can be referenced. At some later time, that VP requesting that page will resume execution and repeat that reference. Going first to ATU 10228, that VP will
40 take an ATU 10228 fault since VP 10212 has not yet been updated to contain that page. The VP will then go to MHT 10716 and MFT 10718 for the required information and, concurrently, WSM 10720 and ATU 10228 will be updated.

In regard to the above operations, each VP active in CS 10110 is assigned a Page Fault Frequency Time Factor (PFFT) which is used by CS 10110's VMM to adjust that VP's working set so that the interval between
45 successive page faults for that VP lies in an optimum time range. This assists in ensuring CS 10110's VMM is operating most efficiently and allows CS 10110's VMM to be tuned as required.

The above discussions have assumed that the page being referenced, whether from a UID/O address, an AON/O address, or a Name, is resident in an object active in CS 10110. While an object need not have a page in MEM 10112 to be active, the object must be active to have a page in MEM 10112. A VP, however,
50 may reference a page in an object not active in CS 10110. If such a reference is made, the object must be made active in CS 10110 before the page can be brought into MEM 10112. The result is an operation similar to the page fault operation described above. CS 10110 maintains an Active Object Manager (AOM), including Active Object Request Queue (AORQ) 10728, which are similar in operation to CS 10110's VMM and VMMRQ 10722. CS 10110's AOM and AORQ 10728 operate in conjunction with AOTHT 10710 and AOT
55 10712 to locate inactive objects and make them active by assigning them AON's and generating entries for them in AOTHT 10710, AOT 10712, and AOTA 10714.

Before a particular object can be made active in CS 10110, it must first be located in backing store (ED 10124). All objects on backing store are located through a Logical Allocation Unit Directory (LAUD) 10730, which is resident in backing store. An LAUD 10730 contains entries for each object accessible to
60 the particular CS 10110. Each LAUD 10730 entry contains the information necessary to generate an AOT 10712 entry for that object. An LAUD 10730 is accessed through a UID/O address contained in CS 10110's VMM. A reference to an LAUD 10730 results in MEM 10112 frames being assigned to that LAUD 10730, and LAUD 10730 being transferred into MEM 10112. If an LAUD 10730 entry exists for the referenced inactive object, the LAUD 10730 entry is transferred into AOT 10712. At the next reference to a page in that object, AOT
65 10712 will provide the AON for that object but, because the page has not yet been transferred into MEM

10112, a page fault will occur. This page fault will be handled in the manner described above and the referenced page transferred into MEM 10112.

Having briefly described the structure and operation of CS 10110's Addressing Structure, including the relationship between UIDs, Names, AONs, and Physical Addresses and the mechanisms by which CS 10110 manages the available address space of MEM 10112, CS 10110's protection structures will be described next below.

E. CS 10110 Protection Mechanisms (Fig. 109)

Referring to Fig. 109, a schematic representation of Protection Mechanisms 10230 is shown. Protection Tables 10232 include Active Primitive Access Matrix (APAM) 10910, Active Subject Number Hash Table (ASNHT) 10912, and Active Subject Table (AST) 10914. Those portions of Protection Mechanism 10230 resident in FU 10120 include ASN Register 10916 and Protection Cache (PC) 10234.

As previously discussed, access rights to objects are arbitrated on the basis of subjects. A subject has been defined as a particular combination of a principle, Process, and Domain (PPD), each of which is identified by a corresponding UID. Each object has associated with it an Access Control List (ACL) 10918 containing an ACL Entry (ACLE) for each subject having access rights to that object.

When an object becomes active in CS 10110 (i.e., is assigned an AON) each ACLE in that object's ACL 10918 is written into APAM 10910. Concurrently, each subject having access rights to that object, and for which there is an ACLE in that object's ACL 10918, is assigned an Active Subject Number (ASN). These ASNs are written into ASNHT 10912 and their corresponding PPDs are written into AST 10914. Subsequently, the ASN of any subject requesting access to that object is obtained by hashing the PPD of that subject to obtain a PPD index into ASNHT 10912. ASNHT 10912 will in turn provide a corresponding ASN. An ASN may be used as an index into AST 10914. AST 10914 will provide the corresponding PPD, which may be compared to an original PPD to confirm the accuracy of the ASN.

As described above, APAM 10910 contains an ACL 10918 for each object active in CS 10110. The access rights of any particular active subject to a particular active object are determined by using that subject's ASN and that object's AON as indexes into APAM 10910. APAM 10910 in turn provides a 4 bit output defining whether that subject has Read (R) Write (W) or Execute (E) rights with respect to that object, and whether that particular entry is Valid (V).

ASN Register 10916 and PC 10234 are effectively acceleration mechanisms of Protection Tables 10232. ASN Register 10916 stores the ASN of a currently active subject while PC 10234 stores certain access right information for objects being used by the current process. PC 10234 entries are indexed by ASNs from ASN register 10916 and by a mode input from JP 10114. Mode input defines whether the current procedure intends to read, write, or execute with respect to a particular object having an entry in PC 10234. Upon receiving ASN and mode inputs, PC 10234 provides a go/nogo output indicating whether that subject has the access rights required to execute the intended operation with respect to that object.

In addition to the above mechanism, each procedure to which arguments may be passed in a cross-domain call has associated with it an Access Information Array (AIA) 10352, as discussed with reference to Virtual Processes 10212. A procedure's AIA 10352 states what access rights a calling procedure (subject) must have to a particular object (argument) before the called procedure can operate on the passed argument. CS 10110's protection mechanisms compare the calling procedure's access rights to the rights required by the called procedure. This insures the calling procedure may not ask a called procedure to do what the calling procedure is not allowed to do. Effectively, a calling procedure can pass to a called procedure only the access rights held by the calling procedure.

Finally, PC 10234, APAM 10910, or AST 10914 faults (i.e., misses) are handled in the same manner as described above with reference to page faults in discussion of CS 10110's Addressing Mechanisms 10220. As such, the handling of protection misses will not be discussed further at this point.

Having briefly described structure and operation of CS 10110's Protection Mechanisms 10230, CS 10110's Micro-Instruction Mechanisms 10236 will be described next below.

F. CS 10110 Micro-Instruction Mechanism (Fig. 110)

As previously described, CS 10110 is a multiple language machine. Each program written in a high level user language is compiled into a corresponding S-Language program containing instructions expressed as S-Interpreters (SINTs) for each S-Language. CS 10110 provides a set, or dialect, of microcode instructions, referred to as S-Interpreters (SINTs) for each S-Language. SINTs interpret S-Interpreters and provide corresponding sequences of microinstructions for detailed control of CS 10110.

Referring to Fig. 110, a partial schematic representation of CS 10110's Micro-Instruction Mechanisms 10236 is shown. At system initialization all CS 10110 microcode, including SINTs and all machine assist microcode, is transferred from backing store to Micro-Code Control Store (mCCS) 10238 in MEM 10112. The Micro-Code is then transferred from mCCS 10238 to FU Micro-Code Structure (FUmC) 10240 and EU Micro-Code Structure (EUmC) 10242. EUmC 10242 is similar in structure and operation to FUmC 10240 and thus will be described in following detailed descriptions of CS 10110's structure and operation. Similarly, CS 10110 machine assist microcode will be described in following detailed discussions. The present discussion will concern CS 10110's S-Interpreter mechanisms.

CS 10110's S-Interpreters (SINTs) are loaded into S-Interpreter Table (SITT) 11012, which is represented

in Fig. 110 as containing S-Interpreters 1 to N. Each SIT contains one or more sequences of micro-code; each sequence of microcode corresponds to a particular SIN in that S-Language dialect. S-Interpreter Dispatch Table (SDT) 11010 contains S-Interpreter Dispatchers (SDs) 1 to N. There is one SD for each SINT in SITT 11012, and thus a SD for each S-Language dialect. Each SD comprises a set of pointers. Each pointer
 5 in a particular SD corresponds to a particular SIN of that SD's dialect and points to the corresponding sequence of microinstructions for interpreting that SIN in that dialect's SIT in SITT 11012. In illustration, as previously discussed when a particular procedure is being executed the SIP for that procedure is transferred into one of mCR's 10366. That SIP points to the start of the SD for the SIT which is to be used to interpret the SINs of that procedure. In Fig. 110, the SIP in mCRs 10366 is shown as pointing to the start of
 10 SD2. Each S-Op appearing during execution of that procedure is an offset, relative to the start of the selected SD, pointing to a corresponding SD pointer. That SD pointer in turn points to the corresponding sequence of microinstructions for interpreting that SIN in the corresponding SIT in SITT 11012. As will be described in following discussions, once the start of a microcode sequence for interpreting an SIN has been selected, CS 10110 micromachine then proceeds to sequentially call the microinstructions of that sequence
 15 from SITT 11012 and use those microinstructions to control operation of CS 10110.

G. Summary of Certain CS 10110 Features and Alternate Embodiments

The above Introductory Overview has described the overall structure and operation and certain features of CS 101, that is, CS 10110. The above Introduction has further described the structure and
 20 operation and further features of CS 10110 and, in particular, the physical implementation and operation of CS 10110's information, control, and addressing mechanisms. Certain of these CS 10110 features are summarized next below to briefly state the basic concepts of these features as implemented in CS 10110. In addition, possible alternate embodiments of certain of these concepts are described.

First, CS 10110 is comprised of a plurality of independently operating processors, each processor
 25 having a separate microinstruction control. In the present embodiment of CS 10110, these processors include FU 10120, EU 10122, MEM 10112 and IOS 10116. Other such independently operating processors, for example, special arithmetic processors such as an array processor, or multiple FU 10120's, may be added to the present CS 10110.

In this regard, MEM 10112 is a multiport processor having one or more separate and independent ports
 30 to each processor in CS 10110. All communications between CS 10110's processors are through MEM 10112, so that MEM 10112 operates as the central communications node of CS 10110, as well as performing memory operations. Further separate and independent ports may be added to MEM 10112 as further processors are added to CS 10110. CS 10110 may therefore be described as comprised of a plurality of separate, independent processors, each having a separate microinstruction control and having a separate
 35 and independent port to a central communications and memory node which in itself is an independent processor having a separate and independent microinstruction control. As will be further described in a following detailed description of MEM 10112, MEM 10112 itself is comprised of a plurality of independently operating processors, each performing memory related operations and each having a separate microinstruction control. Coordination of operations between CS 10110's processors is achieved by
 40 passing "messages" between the processors, for example, SOP's and descriptors.

CS 10110's addressing mechanisms are based, first, upon UID addressing of objects. That is, all information generated for use in or by operation of a CS 10110, for example, data and procedures, is structured into objects and each object is assigned a permanent UID. Each UID is unique within a particular
 45 CS 10110 and between all CS 10110's and is permanently associated with a particular object. The use of UID addressing provides a permanent, unique addressing means which is common to all CS 10110's, and to other computer systems using CS 10110's UID addressing.

Effectively, UID addressing means that the address (or memory) space of a particular CS 10110 includes the address space of all systems, for example disc drives or other CS 10110s, to which that particular CS 10110 has access. UID addressing allows any process in any CS 10110 to obtain access to any
 50 object in any CS 10110 to which it has physical access, for example, another CS 10110 on the other side of the world. This access is constrained only by CS 10110's protection mechanism. In alternate embodiments of CS 10110, certain UIDs may be set aside for use only within a particular CS 10110 and may be unique only within that particular CS 10110. These reserved UIDs would, however, be a limited group known to all CS 10110 systems as not having uniqueness between systems, so that the unique object addressing
 55 capability of CS 10110's UID addressing is preserved.

As previously stated, AONs and physical descriptors are presently used for addressing within a CS 10110, effectively as shortened UIDs. In alternate embodiments of CS 10110, other forms of AONs may be used, or AONs may be discarded entirely and UIDs used for addressing within as well as between CS
 60 10110s.

CS 10110's addressing mechanisms are also based upon the use of descriptors within and between CS
 10110s.

Each descriptor includes an AON or UID field to identify a particular object, an offset field to specify a bit granular offset within the object, and a length field to specify a particular number of bits beginning at the specified offset. Descriptors may also include a type, or format field identifying the particular format of the
 65 data referred to by the descriptor. Physical descriptors are used for addressing MEM 10112 and, in this

case, the AON or UID field is replaced by a frame number field referring to a physical location in MEM 10112.

As stated above, descriptors are used for addressing within and between the separate, independent processors (FU 10120, EU 10122, MEM 10112, and IOS 10116) comprising CS 10110, thereby providing common, system wide bit granular addressing which includes format information. In particular, MEM 10112 responds to the type information fields of descriptors by performing formatting operations to provide requestors with data in the format specified by the requestor in the descriptor. MEM 10112 also accepts data in a format specified in a descriptor and reformats that data into a format most efficiently used by MEM 10112 to store the data.

As previously described, all operands are referred to in CS 10110 by Names wherein all Names within a particular S-Language dialect are of a uniform, fixed size and format. A K value specifying Name size is provided to FU 10120, at each change in S-Language dialect, and is used by FU 10120 in parsing Names from the instruction stream. In an alternate embodiment of CS 10110, all Names are the same size in all S-Language dialects, so that K values, and the associated circuitry in FU 10120's parser, are not required.

Finally, in descriptions of CS 10110's use of SOPs, FU 10120's microinstruction circuitry was described as storing one or more S-Interpreters. S-Interpreters are sets of sequences of microinstructions for interpreting the SOPs of various S-Language dialects and providing corresponding sequences of microinstructions to control CS 10110. In an alternate embodiment of CS 10110, these S-Interpreters (SITT 11012) would be stored in MEM 10112. FU 10120 would receive SOPs from the instruction stream and, using one or more S-Interpreter Base Pointers (that is, architectural base pointers pointing to the SITT 11012 in MEM 10112), address the SITT 11012 stored in MEM 10112. MEM 10112 would respond by providing, from the SITT 11012 in MEM 10112, sequences of microinstructions to be used directly in controlling CS 10110. Alternately, the SITT 11012 in MEM 10112 could provide conventional instructions usable by a conventional CPU, for example, Fortran or machine language instructions. This, for example, would allow FU 10120 to be replaced by a conventional CPU, such as a Data General Corporation Eclipse®.

Having briefly summarized certain features of CS 10110, and alternate embodiments of certain of these features, the structure and operation of CS 10110 will be described in detail below.

2. DETAILED DESCRIPTION OF CS 10110 MAJOR SUBSYSTEMS

(Figs. 201—206, 207—274)

Having previously described the overall structure and operation of CS 10110, the structure and operation of CS 10110's major subsystems will next be individually described in further detail. As previously discussed, CS 10110's major subsystems are, in the order in which they will be described, MEM 10112, FU 10120, EU 10122, IOS 10116, and DP 10118. Individual block diagrams of MEM 10112, FU 10120, EU 10122, IOS 10116, and DP 10118 are shown in, respectively, Figures 201 through 205. Figures 201 through 205 may be assembled as shown in Fig. 206 to construct a more detailed block diagram of CS 10110 corresponding to that shown in Fig. 101. For the purposes of the following descriptions, it is assumed that Figs. 201 through 205 have been assembled as shown in Fig. 206 to construct such a block diagram. Further diagrams will be presented in following descriptions as required to convey structure and operation of CS 10110 to one of ordinary skill in the art.

As previously described, MEM 10112 is an intelligent, prioritizing memory having separate and independent ports MIO 10128 and MJP 10140 to, respectively, IOS 10116 and JP 10114. MEM 10112 is shared by and is accessible to both JP 10114 and IOS 10116 and is the primary memory of CS 10110. In addition, MEM 10112 is the primary path for information transferred between the external world (through IOS 10116) and JP 10114.

As will be described further below, MEM 10112 is a two-level memory providing fast access to data stored therein. MEM 10112 first level is comprised of a large set of random access arrays and MEM 10112 second level is comprised of a high speed cache whose operation is generally transparent to memory users, that is JP 10114 and IOS 10116. Information stored in MEM 10112, in either level, appears to be bit addressable to both JP 10114 and IOS 10116. In addition, MEM 10112 presents simple interfaces to both JP 10114 and IOS 10116. Due to a high degree of pipe lining (concurrent and overlapping memory operations) MEM 10112 interfaces to both JP 10114 and IOS 10116 appear as if each HP 10114 and IOS 10116 have full access to MEM 10112. This feature allows data transfer rates of up to, for example, 63.6 megabytes per second from MEM 10112 and 50 megabytes per second to MEM 10112.

In the following descriptions, certain terminology used on those descriptions will be introduced first, followed by description of MEM 10112 physical organization. Then MEM 10112 port structures will be described, followed by descriptions of MEM 10112's control organization and control flow. Next, MEM 10112's interfaces to JP 10114 and IOS 10116 will be described. Following these overall descriptions the major logical structures of MEM 10112 will be individually described, starting at MEM 10112's interfaces to JP 10114 and IOS 10116 and proceeding inwardly to MEM 10112's first (or bulk) level of data stored. Finally, certain features of MEM 10112 microcode control structure will be described.

A. MEM 10112 (Figs. 201, 206, 207—237)

a. Terminology

Certain terms are used throughout the following descriptions and are defined here below for reference

by the reader.

A word is 32 bits of data

A byte is 8 bits of data

A block is 128 bits of data (that is, 4 words).

5 A block is always aligned on a block boundary, that is the low order 7 bits of logical or physical address are zero (see Chapter 1, Sections A.f and D. Descriptions of CS 10110 Addressing).

The term aligned refers to the starting bit address of a data item relative to certain address boundaries. A starting bit address is block aligned when the low order 7 bits of starting bit address are equal to zero, that is the starting bit address falls on a boundary between adjacent blocks. A word align starting bit address means that the low order 5 bits of starting bit address are zero, the starting bit address points to a boundary between adjacent words. A byte aligned starting bit address means that the low order 3 bits of starting bit address are zero, the starting bit address points to a boundary between adjacent bytes.

10 Bit granular data has a starting bit address falling within a byte, but not on a byte boundary, or the address is aligned on a byte boundary but the length of the data is bit granular, that is not a multiple of 8 bits.

15 b. MEM 10112 Physical Structure (Fig. 201)

Referring to Fig. 201, a partial block diagram of MEM 10112 is shown. Major functional units of MEM 10112 are Main Store Bank (MSB) 20110, including Memory Arrays (MA's) 20112, Bank Controller (BC) 20114, Memory Cache (MC) 20116, including Bypass Write File (BYF) 20118, Field Isolation Unit (FIU) 20120, and Memory Interface Controller (MIC) 20122.

MSB 20110 comprises MEM 10112's first or bulk level of storage. MSB 20110 may include from one to, for example, 16 MA 20112's. Each MA 20112 may have a storage capacity, for example, 256 K-byte, 512 K-byte, 1 M-byte, or 2 M-bytes of storage capacity. As will be described further below, MA 20112's of different capacities may be used together in MSB 20110. Each MA 20112 has a data input connected in parallel to Write Data (WD) Bus 20124 and a data output connected in parallel to Read Data (RD) Bus 20126. MA's 20112 also have control and address ports connected in parallel to address and control (ADCTL) Bus 20128. In particular, Data Inputs 20124 of Memory Arrays 20112 are connected in parallel to Write Data (WD) Bus 20126, and Data Outputs 20128 of Memory Arrays 20112 are connected in parallel to Read Data (RD) Bus 20130. Control Address Ports 20132 of Memory Arrays 20112 are connected in parallel to Address and Control (ADCTL) Bus 20134.

Data Output 20136 of Bank Controller 20114 is connected to WD Bus 20126 and Data Input 20138 of BC 20114 is connected to RD Bus 20130. Control and Address Port 20140 of BC 20114 is connected to ADCTL Bus 20134. BC 20114's Data Input 20142 is connected to MC 20116's Data Output 20144 through Store Back Data (SBD) Bus 20146. BC 20114's Store Back Address Input 20148 is connected to MC 20116 Store Back Address Output 20150 through Store Back Address (SBA) Bus 20152. BC 20114's Read Data Output 20154 is connected to MC 20116's Read Data Input 20156 through Read Data Out (RDO) Bus 20158. BC 20114's Control Port 20160 is connected to Memory Control (MCNTL) Bus 20164.

MC 20116 has Output 20166 connected to MIO Bus 10131 through MIO Port 10128, and Port 20168 connected to MOD Bus 10144 through MJP Port 10140. Control Port 20170 of MC 20116 is connected to MCNTL Bus 20164. Input 20172 of BYF 20118 is connected to IOM Bus 10130 through MIO Port 10128, and Output 20176 is connected to SBD Bus 20146 through Bypass Write In (BWI) Bus 20178.

40 Finally, FIU 20120 has an Output 20180 and an Input 20182 connected to, respectively, MIO Bus 10129 and IOM Bus 10130 through MIO Port 10128. Input 20184 and Port 20186 are connected to, respectively, JPD Bus 10142 and MOD Bus 10144 through MJP Port 10140. Control Port 20188 is connected to MCNTL Bus 20164. Referring finally to MIC 20122, MIC 20122 has Control Port 20190 and Input 20192 connected to, respectively, IOMC Bus 10131 and IOM Bus 10130 through MIO Port 10128. Control Port 20194 and Input 20196 are connected, respectively, to JPMC Bus 10147 and Physical Descriptor (PD) Bus 10146 through MJP Port 10140. Control Port 20198 is connected to MCNTL Bus 20164.

50 c. MEM 10112 General Operation

Referring first to MEM 10112's interface to IOS 10116, this interface includes MIO Bus 10129, IOM Bus 10130, and IOMC Bus 10131. Read and Write Addresses and data to be written into MEM 10112 are transferred from IOS 10116 to MEM 10112 through IOM Bus 10130. Data read from MEM 10112 is transferred to IOS 10116 through MIO Bus 10129. IOMC 10131 is a Bi-directional Control bus between MEM 10112 and IOS 10116 and, as described further below, transfers control signals between MEM 10112 and IOS 10116 to control transfer of data between MEM 10112 and IOS 10116.

MEM 10112's interface to JP 10114 is MJP Port 10140 and includes JPD Bus 10142, MOD Bus 10144, PD Bus 10146, and JPMC Bus 10147. Physical descriptors, that is MEM 10112 physical read and write addresses, are transferred from JP 10114 to MEM 10112 through PD Bus 10146. S Ops, that is sequences of S Instructions and operand names, are transferred from MEM 10112 to JP 10114 through MOD Bus 10144 while data to be written into MEM 10112 from JP 10114 is transferred from JP 10114 to MEM 10112 through JPD Bus 10142. JPMC Bus 10147 is a Bi-directional Control bus for transferring command and control signals between MEM 10112 and JP 10114 for controlling transfer of data between MEM 10112 and JP 10114. As will be described further below, MJP Port 10140, and in particular MOD Bus 10144 and PD Bus

10146, is generally physically organized as a single port that operates as a dual port. In a first case, MJP Port 10140 operates as a Job Processor Instruction (JI) Port for transferring S Ops from MEM 10112 to JP 10114. In a second case, MOD 10144 and PD 10146 operate as a Job Processor Operand (JO) Port for transfer of operands, from MEM 10112 to JP 10114, while JPD Bus 10142 and PD Bus transfer operands from JP 10114 to MEM 10112.

Referring to MSB 20110, MSB 20110 contains MEM 10112's first, or bulk, level of storage capacity. MSB 20110 may contain from one to, for example, 16 MA's 20112. Each MA 20112 contains a dynamic, random access memory array and may have a storage capacity of, for example 256 Kilo-bytes, 512 Kilo-bytes, 1 Mega-bytes, or 2 Mega-bytes. MEM 10112 may therefore have a physical capacity of up to, for example, 16 Mega-bytes of bulk storage. As will be described further below, MA 20112's of different capacity may be used together in MSB 20110, for example, four 2 Mega-byte MA 20112's and four 1 Megabyte MA 20112's.

BC 20114 controls operation of MA's 20112 and is the path for transfer of data to and from MA's 20112. In addition, BC 20114 performs error detection and correction on data transferred into and out of MA's 20112, refreshes data stored in MA's 20112, and, during a refresh operations, performs error detection and correction of data stored in MA's 20112.

MC 20116 comprises MEM 10112's second, or cache, level of storage capacity and contains, for example 8 Kilo-bytes of high speed memory. MC 20116, including BYF 20118, is also the path for data transfer between MSB 20110 (through BC 20114) and JP 10114 and IOS 10116. In general, all read and write operations between JP 10114 and IOS 10116 are through MC 20116. IOS 10116 may, however, perform read and write operations of complete blocks by-passing MC 20116. Block write operations from IOS 10116 are accomplished through BYF 20118 while block read operations are performed through a data transfer path internal to MC 20116 and shown and described below. All read and write operations between MEM 10112 and JP 10114, however, must be performed through the cache internal to MC 20116, as will be shown and described further below.

As also shown and described below, FIU 20120 includes write data registers for receiving data to be written into MEM 10112 from JP 10114 and IOS 10116, and circuitry for manipulating data read from MSB 20110 so that MEM 10112 appears as a bit addressable memory. FIU 20120, in addition to providing bit addressability of MEM 10112, performs right and left alignment of data, zero fill of data, sign extension operations, and other data manipulation operations described further below. In performing these data manipulation operations on data read from MEM 10112 to JP 10114, MOD Bus 10144 is used as a data path internal to MEM 10112 for transferring of data from MC 20116 to FIU 20120, and from FIU 20120 to MC 20116. That is, data to be transferred to JP 10114 is read from MC 20116, transferred through MOD Bus 10144 to FIU 20120, manipulated by FIU 20120, and transferred from FIU 20120 to JP 10114 through MOD Bus 10144.

MIC 20122 contains circuitry controlling operation of MEM 10112 and, in particular, controls MEM 10112's interface with JP 10114 and IOS 10116. MIC 20122 receives MEM 10112 read and write request, that is read and write addresses through PD Bus 10146 and IOM Bus 10130 and control signals through JPMC Bus 10147 and IOMC Bus 10131, and provides control signals to BC 20114, MC 20116, and FIU 20120 through MCNTL Bus 20164.

Having described the overall structure and operation of MEM 10112, the structure and operation of MEM 10112's Port, MIO Port 10128, and MJP Port 10140, will be described next, followed by descriptions of MEM 10112's control structure and the control and flow of MEM 10112 read and write requests.

d. MEM 10112 Port Structure

MEM 10112 port structure is designed to provide a simple interface to JP 10114 and IOS 10116. While providing fast and flexible operation in servicing MEM 10112 read and write requests from JP 10114 and IOS 10116. In this regard, MEM 10112, as will be described further below, may handle up to 4 read and write requests concurrently and up to, for example, a 63.6 M-byte per second data rate. In addition MEM 10112 is capable of performing bit granular addressing, block read and write operations, and data manipulations, such as alignment and filling, to enable JP 10114 and IOS 10116 to operate most efficiently.

MEM 10112 effectively services requests from three ports. These ports are MIO Port 10128 to IOS 10116, hereafter referred to as IO Port, and JI and JO Ports, described above, to JP 10114. These three ports share the entire address base of MEM 10112, but IOS 10116, for example, may be limited from making full use of MEM 10112's address space. Each port has a different set of allowed operations. For example, JO Port can use a bit granular addresses but can reference only 32 bits of data on each request. JI Port can make read requests only to word align 32 bit data items. IO Port may reference bit granular data, and, as described further below, may read or write up to 16 bytes on each read or write request. The characteristics of each of these ports will be discussed next below.

1. IO Port Characteristics

IOS 10116 may access MEM 10112 in either of two modes. The first mode is block transfers by-passing or through the cache in MC 20116, and the second is non-block transfer through the cache and MC 20116.

Block by-passes may occur for both read and write operations. A read or write operation is eligible for a block by-pass if the data is on block boundaries, is 16 bytes long, and the read or write request is not accompanied by a control signal indicating that an encache (load into MC 20116's cache) operation is to be

EP 0 067 556 B1

performed. A by-pass operation takes place only if the block address, that is the physical address of the block in MEM 10112 does not address a currently encached block, that is the block is not present in MC 20116's cache. If the block is encached in MC 20116's cache, the read or write transfer is to MC 20116's cache.

5 Partial block references, that is non-full block transfers will go through MC 20116's cache. If a cache miss occurs, that is the reference data is not present in MC 20116's cache, MEM 10112's control structures transfer the data to or from MSB 20110 and update MC 20116's cache. It should be noted that partial blocks may be as short as one byte, or up to 15 bytes long. A starting byte address may be anywhere within a block, but the partial block's length may not cross a block boundary.

10 Bit length transfers, that is transfers of data items having a length of 1 to 16 bits and not a multiple of a byte, or where address is not on a byte boundary, go through MC 20116's cache. These operations may cross byte, word, or block boundaries but may not cross page boundaries. These specific operations requested by IO port determines whether a read or write request is a partial block or bit length transfer.

15 2. JO Port Characteristics

All read or write requests from JO Port must go through MC 20116's cache; by-pass operations may not be performed. The data transferred between MEM 10112 and JP 10114 is always 32 bits in length but, of the 32 bits passed, from zero to 32 bits may be valid data. JP 10114 determines the location of valid data within the 32 bits by referring to certain FIU specification bits provided as part of the read or write request. As will be described further below, FIU specification bits, and other control bits, are provided to MIC 20122 by JP 10114 through JPMC Bus 10147 when each read or write request is made.

While MEM 10112 does not perform block by-pass operations to JP 10114, MEM 10112 may perform a cache read-through operation. Such operations occur on a JP 10114 read request wherein the requested data is not present in MC 20116's cache. If the JP 10114 read request is for a full word, which is word aligned, MEM 10112's Load Manager, discussed below, transfers the requested data directly to JP 10114 while concurrently loading the requested data into MC 20116's cache. This operation is referred to as a "hand-off" operation. These operations may also be performed by IO Port for 16 bit half words aligned on the right hand half word of a 32 bit word, or if a full block is handed left and loaded into MC 20116's cache.

30 3. JI Port Characteristics

All JI Port requests are satisfied through MC 20116's cache; MEM 10112 does not perform by-pass operations to JI Port. JI Port requests are always read requests for full-word aligned words and are handed off, as described above, if a cache miss occurs. In most other respects, JI Port requests are similar to JO Port requests.

35 Having described the overall structure and operation of MEM 10112, including MEM 10112's input and output ports to JP 10114 and IOS 10116, MEM 10112's control structure will be described next below.

e. MEM 10112 Control Structure and Operation (Fig. 207)

40 Referring to Fig. 207, a more detailed block diagram of MIC 20116 is shown. Fig. 207 will be referred to in conjunction with Fig. 201 in the following discussion of MEM 10112's control structure.

1. MEM 10112 Control Structure

45 Referring first to Fig. 207, MCNTL Bus 20164 is represented as including MCNTL-BC Bus 20164A, MCNTL-MC Bus 20164B, and MCNTL-FIU Bus 20164C. Buses 20164A, 20164B, and 20164C are branches of MCNTL Bus 20164 connected to, respectively, BC 20114, MC 20116, and FIU 20120. Also represented in Fig. 207 are PD Bus 10146 and JPMC Bus 10147 to JP 10114, and IOM Bus 10130 and IOMC Bus 10131 to IOS 10116.

50 JO Port Address Register (JOPAR) 20710 and JI Port Address Register (JIPAR) 20712 have inputs connected from PD Bus 10146. IO Port Address Register (IOPAR) 20714 has an input connected from IOM Bus 10130. Port Control Logic (PC) 20716 has a bi-directional input/outputs connected from JPC 10147 and IOMC Bus 10131. By-pass Read/Write Control Logic (BR/WC) 20718 has a bidirectional input/output connected from IOMC Bus 10131.

55 Outputs of JOPAR 20710, JIPAR 20712, and IOPAR 20714 are connected to inputs of Port Request Multiplexer (PRMUX) 20720 through, respectively, Buses 20732, 20734, 20736. PRMUX 20720's output in turn is connected to Bus 20738. Branches of Bus 20738 are connected to inputs of Load Pointers (LP) 20724, Miss Control (MISSC) 20726, and Request Manager (RM) 20722, and to Buses MCNTL-MC 20164B and MCNTL-FIU 20164C.

60 Outputs of PC 20716 are connected to inputs of JOPAR 20710, JIPAR 20712, IOPAR 20714, PRMUX 20720, and LP 20724 through Bus 20738. Bus 20740 is connected between an input/output of PC 20716 and in input/output of RM 20722.

An output of BR/WC 20718 is connected to MCNTL-MC Bus 20164B through Bus 20742. Inputs of BR/WC 20718 are connected from outputs of RM 20722 and Read Queue (RQ) 20728 through, respectively, Buses 20744 and 20746.

65 RM 20722 has outputs connected to MCNTL-BC Bus 20164A, MCNTL-FIU Bus 20164C, and input of MISSC 20726, and an input of LP 20724 through, respectively, Buses 20748, 20750, 20752, and 20754.

EP 0 067 556 B1

MISSC 20726's output is connected to MCNTL-BC Bus 20164A. Outputs of LP 20724 are connected to MCNTL-MC Bus 20164B and to an input of LM 20730 through, respectively, Buses 20756 and 20758. RQ 20728's input is connected from MCNTL-MC Bus 20164B through Bus 20760 and RQ 20728 has outputs connected to an input of LP 20724, through Bus 20762, and as previously described to an input of BR/WC 20718 through Bus 20746. Finally, LM 20730's output is connected to MCNTL-MC Bus 20164B through Bus 20764.

Having described the structure of MIC 20716 with reference to Fig. 207, and having previously described the structure of MEM 10112 with reference to Fig. 201, MEM 10112's control structure operation will next be described with reference to both figures 201 and 207.

2. MEM 10112 Control Operation

Referring first to Fig. 207, JOPAR 20710, JIPAR 20712, and IOPAR 20714 are, as previously described, connected from PD Bus 10146 from JP 10114 and IOM Bus 10130 from IOS 10116. JPAR 20710, JIPAR 20712, and IOPAR 20714 receive read and write request addresses from JP 10114 and IOS 10116 and store these addresses for subsequent service by MEM 10112. As will be described further below, these address inputs from JP 10114 and IOS 10116 include FIU information specifying what data manipulation operations must be performed by FIU 20120 before requested data is transferred to the requestor or written into MEM 10112, information regarding the destination data read from MEM 10112 is to be provided to, information regarding the type of operation to be performed by MEM 10112, and information regarding operand length. Request address information received and stored in JOPAR 20710, JIPAR 20712, and IOPAR 20714 is retained therein until MEM 10112 has initiated service of the corresponding requests. MEM 10112 will accept further request address information into a given port register only after a previous request into that port has been serviced or aborted. Address information outputs from JOPAR 20710, JIPAR 20712, and IOPAR 20714 is transferred through PRMUX 20720 to Bus 20738 and from there to RM 20722, MC 20116, and FIU 20120 as service of individual requests is initiated. As will be described below, this address information will be transferred through PRMUX 20720 and Bus 20738 to LP 20724 for use in servicing a cache miss upon occurrence of a MC 20116 miss.

PC 20716 receives command and control signals pertinent to each requested memory operation from JP 10114 and IOS 10116 through JPMC Bus 10147 and IOSB Bus 10131. PC 20716 includes request arbitration logic and port state logic. Request arbitration logic determines the sequence in which IO, JI, JO ports are serviced, and when each port is to be serviced. In determining the sequence of port service, request arbitration logic uses present port state information for each port from the port state logic, information from JPMC Bus 10147 and IOMC Bus 10131 regarding each incoming request, and information from RM 20722 concerning the present state of operation of MEM 10112. Port state logic selects each particular port to be serviced and, by control signals through Bus 20738, enables transfer of each port's request address information from JOPAR 20710, JIPAR 20712, and IOPAR 20714 through PRMUX 20720 to Bus 20738 for use by the remainder of MEM 10112's control logic in servicing the selected port. In addition to request information received from JP 10114 and IOS 10116 through JPMC Bus 10147 and IOMC Bus 10131, port state logic utilizes information from RM 20722 and, upon occurrence of a cache miss, from LM 20730 (for clarity of presentation, this connection is not represented in Fig. 207). Port state logic also controls various port state flag signals, for example port availability signals, signals indicating valid requests, and signals indicating that various ports are waiting service.

RM 20722 controls execution of service for each request. RM 20722 is a microcode controlled "micromachine" executing programs called for by requested MEM 10112 operations. Inputs of RM 20722 include request address information from IOPAR 20714, JIPAR 20712, and JOPAR 20710, including information regarding the type of MEM 10112 operation to be performed in servicing a particular request, interrupt signals from other MEM 10112 control elements, and, for example, start signals from PC 20716's request arbitration logic. RM 20722 provides control signals to FIU 20120, MC 20116, and most other parts of MEM 10112's control structure.

Referring to Fig. 201, MC 20116's cache is, for example, an 8 Kilo-byte, four set associative cache used to provide rapid access to a subset of data stored in MSB 20110. The subset of MSB 20110 data stored in MC 20116's cache at any time is the data most recently used by JP 10114 or IOS 10116. MC 20116's cache, described further below, includes tag store comparison logic for determining encached addresses, a data store containing corresponding encached data, and registers and logic necessary to up-date cache contents upon occurrence of a cache miss. Registers and logic for servicing cache misses includes logic for determining the least recently used cache entry and registers for capture and storage of information regarding missed cache references, for example modify bits and replacement page numbers. Inputs to MC 20116 are provided from RM 20722, LM 20730 (discussed further below), FIU 20120, MSB 20110 (through BC 20114), LP 20724 (described further below) and address information from PRMUX 20720. Outputs of MC 20116 include data and go to FIU 20120 (through MOD Bus 10144), the data requestors (JP 10114 and IOS 10116), and a MC 20116 Write Back File (described further below).

As previously described, FIU 20120 includes logic necessary to make MEM 10112 appear bit addressable. In addition, FIU 20120 includes logic for performing certain data manipulation operations as required by the requestors (JP 10114 or IOS 10116). Data is transferred into FIU 20120 from MC 20116 through that portion of MOD Bus 10144 internal to MEM 10112, is manipulated as required, and is then

EP 0 067 556 B1

transferred to the requestor through MOD Bus 10144 or MIO Bus 10129. In the case of writes requiring read-modify-write of encached data, the data is transferred back to MC 20116 through MOD Bus 10144 after manipulation. In general, data manipulation operations include locating requested data onto selected MOD Bus 10144 or MIO Bus 10139 lines and filling unused bus lines as specified by the requestor. Data inputs to
5 FIU 20120 may be provided from MC 20116 or JP 10114 through MOD Bus 10144 or from IOS 10116 through IOM Bus 10130. Data outputs from FIU 20120 may be provided to MC 20116, JP 10114, or IOS 10116 through these same buses. Control information is provided to FIU 20120 from RM 20722 through Bus 20748 and MCNTL-FIU Bus 20164C. Address information may be provided to FIU 20120 from JOPAR 20710, JIPAR 20712, or IOPAR 20714 through PRMUX 20720, Bus 20738, and MCNTL-FIU Bus 20164C.

10 Returning to Fig. 207, MISSC 20726 is used in handling MC 20116 misses. In the event of a request referring to data not in MC 20116's cache, MISSC 20726 stores block address of the reference and type of operation to be performed, this information being provided from an address register in MC 20116 and from RM 20722. MISSC 20726 utilizes this information in generating a command to BC 20114, through MCNTL-BC Bus 20164A, for a data read from MSB 20110 to obtain the referenced data. BC 20114 places this
15 command in a queue, or register, and subsequently executes the commanded read operation. MISSC 20726 also generates an entry into RQ 20728 (described further below) indicating the type of operation to be performed when referenced data is subsequently read from MSB 20110.

RQ 20728 is, for example, a three-level deep queue storing information indicating operations associated with data being read from MSB 20110. Two kinds of operation may be indicated: block by-pass reads and cache loads. If a cache load is specified, that is a read and store to MC 20116's cache, is indicated, RM 20722 is interrupted and forced to place other MEM 10112 operations in idle until cache load is completed. A block by-pass read operation results in by-pass read control (described below) assuming control of the data from MSB 20110. Inputs to RQ 20728 are control signals from RM 20752, MISSC 20726, and BC 20114. RQ 20728 provides control outputs to LP 20724 (described below) LM 20730 (described
25 below) RM 20722, and by-pass read control (described below).

LP 20724 is a set of registers for storing information necessary for servicing MC 20116 misses that result in order to load MC 20116's tag store. LM 20730 uses this information when data stored in MSB 20110 and read from MSB 20110 to service a MC 20116 cache miss, becomes available through BC 20114. Inputs to LP 20724 include the address of the missing reference, provided from JOPAR 20710, JIPAR 20712, or
30 IOPAR 20714 through PRMUX 20720 and Bus 20738, commands from RM 20722, and a control signal from RQ 20728. LP 20724 outputs include addresses of missed references to MC 20116, through Bus 20756 and MNCTL-MC 20164B, and command signals to LM 20730 and BR/WC 20718.

LM 20730, referred to above, controls loading of MC 20116's cache with data from MSB 20110 after occurrence of a cache miss. RQ 20728, referred to above, indicates, for each data read from MSB 20110,
35 whether the data read is the result of a MC 20116 cache miss. If the data is read from MSB 20110 as a result of a cache miss, LM 20730 proceeds to issue a sequence of control signals for loading the data from MSB 20110 and its associated address into MC 20116's cache. This data is transferred into MC 20116's cache data store while the block address, from LP 20724 is transferred into the tag store (described in the following discussion) of MC 20116's cache. If the transfer of data into MC 20116's cache replaces data previously
40 resident in that cache, and that previous data is "dirty", that is has been written into so as to be different from an original copy of the data stored on MSB 20110, the modified data resident in MC 20116's cache must be written back into MSB 20110. This operation is performed through a Write Back File contained in MC 20116 and described below. In the event of such an operation, LM 20730 initiates a write back operation by MC 20116 and BC 20114, also as described below.

45 As will be described further in a following description, all MC 20116 cache load operations are full 4 word blocks. A request resulting in a MC 20116 cache miss may result in a "hand-off", that is a read operation of a full 4 word block. Handoff operations also may be of single 32 bit words wherein a 32 bit word aligned word is transferred from JP 10114 or a 16 bit operand aligned on the right half-word is transferred from IOS 10116. In such a handoff operation, LM 20730 will send a valid request signal to the
50 requesting port and a handoff operation will be performed. Otherwise, a waiting signal will be sent to the requesting port and the request will re-enter the priority queue of PC 20716 for subsequent execution. To accomplish these operations, LM 20730 receives input from RQ 20728, (not shown in Fig. 207 for clarity of presentation) and LP 20724. LM 20730 provides outputs to port state logic of PC 20716, to MC 20116, MC 20116's Write Back File and MC 20116's Write Back Address Register and to BC 20114.

55 Referring to Fig. 201, as previously discussed IOS 20116 may request a full block write operation directly to MSB 20110. Such a by-pass write request may be honored if the block being transferred is not encached in MC 20116's cache. In such a case, RM 20722 will initiate the transfer setting up By-pass Write Control logic in BR/WC 20718, and may then pass control of the operation over to BR/WC 20718's By-Pass Write Control logic for completion. By-pass Write Control may then accept the remaining portion of the data block from IOS 10116, generating appropriate hand shaking signals through IOMC Bus 10131, and
60 load the data block into BYF 20118 and MC 20116. MISSC 20726 will provide a by-pass write command to BC 20114, through MNCTL-PC Bus 20164A. BC 20114 will then transfer the data block from BYF 20118 and into MA's 20112 and MSB 20110.

As previously described, BYF 20118 receives data from IOM Bus 10130 and provides data output to BC
65 20114 through BWY Bus 20178 and SBD Bus 20146. BYF 20118 is capable of simultaneously accepting data

EP 0 067 556 B1

from IOM Bus 10130 while reading data out to BC 20114. Control of writing data into BYF 20118 is provided from BRWC 20718's By-Pass Write Control logic.

IOS 10116 may, as previously described, request a full block read operation by-passing MC 20116's cache. In such a case, BRWC 20718's by-pass read control handles data transfer to IOS 10116 and generates required hand shaking signals to IOS 10116 through IOMC Bus 10131. The data path for by-pass read operations is through a data path internal to MC 20116, rather than through BYF 20118. This internal data path is RDO Bus 20158 to MIO Bus 10129.

As previously described, BC 20114 manages all data transfers to and from MA's 20112 in MSB 20110. BC 20114 receives requests for data transfers from RM 20722 in an internal queue register. All data transfers to and from MSB 20110 are full block transfers with block aligned addresses. On data write operations, BC 20114 receives data from BWF 20118 or from MC 20116's Write Back File and transfers the data into MA's 20112. During read operations, BC 20114 fetches the data block from MA's 20112 and places the data block on RDO Bus 20158 while signalling to MIC 20122 that the data is available. As described above, MIC 20122 tracks and controls transfer of data and BYF 20118, MC 20116, and MC 20116's Write Back File, and directs data read from MSB 20110 to the appropriate destination, MC 20116's Data Store, JP 10114, or IOS 10116.

In addition to the above operations, BC 20114 controls refresh of MA's 20112 and performs error detection and correction operations. In this regard, BC 20114 performs two error detection and correction operations. In the first, BC 20114 detects single and double bit errors in data read from MSB 20110 and corrects single bit errors. In the second, BC 20114 reads data stored in MA's 20112 during refresh operations and performs single bit error detection. Whenever an error is detected, during either read operations or refresh operations, BC 20114 makes a record of that error in an error log contained in BC 20114 (described further in a following description). Both JP 10114 and IOS 10116 may read BC 20114's error log, and information from BC 20114's error log may be recorded in a CS 10110 maintenance log and to assist in repair and trouble shooting of CS 10110. BC 20114's error log may be addressed directly by RM 20722 and data from BC 20114's error log is transferred to JP 10114 or IOS 10116 in the same manner as data stored in MSB 20110.

Referring finally to MA's 20112, each MA 20112 contains an array of dynamic semiconductor random access memories. Each MA 20112 may contain 256 Kilo-bytes, 512 Kilo-bytes, 1 Mega-bytes, or 2 Mega-bytes of data storage. The storage capacity of each MA 20112 is organized as segments of 256 Kilo-bytes each. In addressing a particular MA 20112, BC 20114 selects that particular MA 20112 as will be described further below. BC 20114 concurrently selects a segment within that MA 20112, and a block of four words within that segment. Each word may comprise 39 bits of information, 32 bits of data and 7 bits of error correcting code. The full 39 bits of each MA 20112 word are transferred between BC 20114 and MA's 20112 during each read and write operation. Having briefly described the general structure and operation of MEM 10112, certain types of operations which may be performed by MEM 10112 will be described next below.

f. MEM 10112 Operations

MEM 10112 may perform two general types of operation. The first type are data transfer operations and the second type are memory maintenance operations. Data transfer operations may include read, write, and read and set. Memory maintenance operations may include read error log, repair block, and flush cache. Except during a flush cache operation, the existence of MC 20116 and its operation is invisible to the requestors, that is JP 10114 and IOS 10116.

A MEM 10112 read operation transfers data from MS 10112 to a requestor, either JP 10114 or IOS 10116. A read data transfer is asynchronous in that the requestor cannot predict elapsed time between submission of a memory operation request and return of requested data. Operation of a requestor in MEM 10112 is coordinated by a requested data available signal transmitted from MEM 10112 to the requestor.

A MEM 10112 write operation transfers data from either JP 10114 or IOS 10116 to MEM 10112. During such operations, JP 10114 is not required to wait for a signal from MEM 10112 that data provided to MEM 10112 from JP 10114 has been accepted. JP 10114 may transfer data to MEM 10112's JO Port whenever a JO Port available signal from MEM 10112 is present; read data is accepted immediately without further action or waiting required of JP 10114. Word write operations from IOS 10116 are performed in a similar manner. On block write operations, however, IOS 10116 is required to wait for a data taken signal from MEM 10112 before sending the 2nd, 3rd and 4th words of a block.

MEM 10112 has a capability to perform "lock bit" operations. In such operations, a bit granular read of the data is performed and the entire operand is transmitted to the requestor. At the same time, the most significant bit of the operand, that is the Lock Bit, is set to one in the copy of data stored in MEM 10112. In the operand sent to the requestor, the lock bit remains at its previous value, the value before the current read and set operation. Test and set operations are performed by performing read and set operations wherein the data item length is specified to be one bit.

As previously described, MEM 10112 performs certain maintenance operations, including error detection. MEM 10112's Error Log in BC 20114 is a 32 bit register containing an address field and an error code field. On a first error to occur, the error type and in some cases, such as ERCC errors on read data stored in MSB 20110, the address of the data containing the error are stored in BC 20114's Error Log Register. An interrupt signal indicating detection of an error is raised at the same that information

regarding the error is stored in the Error Log. If multiple errors occur before Error Log is read and reset, the information regarding the first error will be retained and will remain valid. The Error Log code field will, however, indicate that more than one error has occurred.

JP 10114 may request a read Error Log operation referred to as a "Read Log and Reset" operation. In this operation, MEM 10112 reads the entire contents of Error Log to JP 10114, resets Error Log Register, and resets the interrupt signal indicating presence of an error. IOS 10116, as discussed further below, is limited to reading 16 bits at a time from MEM 10112. It therefore requires two read operations to read Error Log. First read operation to IOS 10116 reads an upper 16 bits of Error Log data and does not reset Error Log. The second read operation is performed in the same manner as a JP 10114 Read Log and Reset operation, except that only the low order 16 bits of Error Log are read to IOS 10116.

MEM 10112 performs repair block operations to correct parity or ERCC errors in data stored in MC 20116's Cache or in data stored in MA's 20112. In a repair block procedure, parity bits for data stored in MC 20116's Cache, or ERCC check bits of data stored in MA's 20112, are modified to agree with the data bits of data stored therein. In this regard, repaired uncorrectible errors, such as two bit errors of data in MA's 20112, will have good ERCC and parity values. Until a repair block operation is performed, any read request directed to bad data, that is data having parity or ERCC check bits indicating invalid data, will be flagged as invalid. Repair block operations therefore allow such data to be read as valid, for example to be used in a data correction operation. Errors are ignored and not logged in BC 20114's Error Log in repair block operations. A write operation into an area containing bad data may be accomplished if MEM 10112's internal operation does not require a read-modified-write procedure. Only byte aligned writes of integral byte length data residing in MC 20116 and word aligned writes of integral word lengths of data in MSP 20110 do not require read-modified-write operation. By utilizing such write operations, it is therefore possible to overwrite bad data by use of normal write operations before or instead of repair block operations.

MEM 10112 performs a cache flush operation in event of a power failure, that is when MEM 10112 goes into battery back-up operation. In such an event, only MA's 20112 and BC 20114 remain powered. Before JP 10114 and IOS 10116 lose power, JP 10114 and IOS 10116 must transfer to MEM 10112 any data, including operating state, to be saved. This is accomplished by using a series of normal write operations. After conclusion of these write operations, both JP 10114 and IOS 10116 transmit a flush cache request to MEM 10112. Upon receiving two flush cache requests, MEM 10112 flushes MC 20116's Cache so that all dirty data encached in MC 20116's Cache is transferred into MA's 20112 before power is lost. If only JP 10114 or IOS 10116 is operating, DP 10118 will detect this fact and will have transmitted an enabling signal (FLUSHOK) to MEM 10112 during system initialization. FLUSHOK enables MEM 10112 to perform cache flush upon receiving a single flush cache request. After a cache flush operation, no further MEM 10112 operations are possible until Dp 10118 resets a power failure lock-out signal to enable MEM 10112 to resume normal operation.

Having described MEM 10112's overall structure and operation and certain operations which may be performed by MEM 10112, MEM 10112's interfaces to JP 10114 and IOS 10116 will be described next below.

g. MEM 10112 Interfaces to JP 10114 and IOS 10116 (Figs. 209, 210, 211, 204)

As previously described, MJP Port 10140 and MIO Port 10128 logically function as three independent ports. These ports are an IO Port to IOS 10116, a JP Operand Port to JP 10114 and a JP Instruction Port to JP 10114. Referring to Figs. 209, 210, and 211, diagrammatic representations of IO Port 20910, JP Operand (JPO) Port 21010, and JP Instruction (JPI) port 21110 are shown respectively.

IO Port 20910 handles all IOS 10116 requests to MEM 10112, including transfer of both instructions and operands. JPO Port 21010 is used for read and write operations of operands, for example numeric values, to and from JP 10114. JPI Port 21110 is used to read SiNs, that is SOPs and operand NAMEs, from MEM 10112 to JP 10114. Memory service requests to a particular port are serviced in the order that the requests are provided to the Port. Serial order is not maintained between requests to different ports, but ports may be serviced in the order of their priority. In one embodiment of the present invention, IO Port 20910 is accorded highest priority, followed by JPO port 21010, and lastly by JPI Port 21110, with requests currently contained in a port having priority over incoming requests. As described above and will be described in more detail in following descriptions, MEM 10112 operations are pipelined. This pipelining allows interleaving of requests from IO Port 20910, JPO Port 21010, and JPI port 21110, as well as overlapping service of requests at a particular port. By overlapping operations it is meant that one operation servicing a particular port begins before a previous operation servicing that port has been completed.

1. IO Port 20910 Operating Characteristics (Figs. 209, 204)

Referring first to Fig. 209, a diagrammatic representation of IO port 20910 is shown. Signals are transmitted between IO Port 20910 and IOS 10116 through MIO Bus 10129, IOM Bus 10130, and IOMC Bus 10131. MIO Bus 10129 is a unidirectional bus having inputs from MC 20116 and FIU 20120 and dedicated to transfers of data and instructions from MEM 10112 to IOS 10116. IOM Bus 10130 is likewise a unidirectional bus and is dedicated to the transfer, from IOS 10116 to MEM 10112, of read addresses, write addresses, and data to be written into MEM 10112. IOM Bus 10130 provides inputs to BYF 20118, FIU 20120, and MIC 20122. IOMC Bus 10131 is a set of dedicated signal lines for the exchange of control signals between IOS 10116 and MEM

10112.

Referring first to MIO Bus 10129, MIO Bus 10129 is a 36 bit bus receiving read data inputs from MC 20116's Cache and from FIU 20120. A single read operation from MEM 10112 to IOS 10116 transfers one 32 bit word (or 4 bytes) of data (MIO(0—31)) and four bits of odd parity (MIOP(0—3)), or one parity bit per byte.

Referring next to IOM Bus 10130, a single transfer from IOS 10116 to MEM 10112 includes 36 bits of information which may comprise either a memory request comprising a physical address, a true length, and command bits. These memory requests and data are multiplexed onto IOM 10130 by IOS 10116.

Data transfers from IOS 10116 to MEM 10112 each comprise a single 32 bit data word (IOM(0—31)) and four bits of odd parity (IOMP(0—3)) or one parity bit per byte. Such data transfers are received by either BYF 20118 or FIU 20120.

Each IOS 10116 memory request to MEM 10112, as described above, an address field, a length field, and an operation code field. Address and length fields occupy the 32 IOM Bus 10130 lines used for transfer of data to MEM 10112 in IOS 10116 write operations. Length field includes four bits of information occupying bits (IOM(03)) of IOM Bus 10130 and address field contains 27 bits of information occupying bits (IOM(4—31)) of IOM Bus 10130. Together, address and length field specify a physical starting address and true length of the particular data item to be written into or read from MEM 10112. Operation code field specifies the type of operation to be performed by MEM 10112. Certain basic operation codes comprise 3 bits of information occupying bits (IOMP (32—36)) of IOM Bus 10130; as described above. These same lines are used for transfer of parity bits during data transfers. Certain operations which may be requested of MEM 10112 by IOS 10116 are, together with their corresponding command code fields, are;

000 = read,
 001 = read and set,
 010 = write,
 011 = error,
 100 = read error log (first half),
 101 = read error log (second half) and reset,
 110 = repair block, and
 111 = flush cache.

Two further command bits may specify further operations to be performed by MEM 10112. A first command bit, indicates to MEM 10112 during write operations whether it is desirable to encache the data being written into MEM 10112 in MC 20116's Cache. IOS 10116 may set this bit to zero if reuse of the data is unlikely, thereby indicating to MEM 10112 that MEM 10112 should avoid encaching the data. IOS 10116 may set this bit to one if the data is likely to be reused, thereby indicating to MEM 10112 that it is preferable to encache the data. A second command bit is referred to a CYCLE. CYCLE command bit indicates to MEM 10112 whether a particular data transfer is a single cycle operation, that is a bit granular word, or a four cycle operation, that is a block aligned block or a byte aligned partial block.

IOMC 10131 includes a set of dedicated lines for exchange of control signals between IOS 10116 and MEM 10112 to coordinate operation of IOS 10116 and MEM 10112. A first such signal is Load IO Request (LIOR) from IOS 10116 to MEM 10112. When IOS 10116 wishes to load a memory request into MEM 10112, IOS 10116 asserts LIOR to MEM 10112. IOS 10116 must assert LIOR during the same system cycle during which the memory request, that is address, length, and command code fields, are valid.

If LIOR and IO Port Available (IOPA) signals, described below, are asserted during the same clock cycle, MEM 10112's port is loaded from IOS 10116 and IOPA is dropped, indicating the request has been accepted. If a load of a request is attempted and IOPA is not asserted, MEM 10112 remains unaware of the request, LIOR remains active, and the request must then be repeated when IOPA is asserted.

IOPA is a signal from MEM 10112 to IOS 10116 which is asserted by MEM 10112 when MEM 10112 is available to accept a new request from IOS 10116. IOPA may be asserted while a previous request from IOS 10116 is completing operation if the address, length, and operation code fields of the previous request are no longer required by MEM 10112, for example in servicing bypass operations.

IO Data Taken (TIOMD) is a signal from MEM 10112 to IOS 10116 indicating that MEM 10112 has accepted data from IOS 10116. IOS 10116 places a first data word on IOM Bus 10130 on the next system clock cycle after a write request is loaded; that is, LIOR has been asserted, a memory request presented, and IOPA dropped. MEM 10112 then takes that data word on the clock edge beginning the next system clock cycle. At this point, MEM 10112 asserts TIOMD to indicate the data has been accepted. On a single word operations TIOMD is not used by IOS 10116 as a first data word is always accepted by MEM 10112 if IO Port 20910 was available. On block operations, a first data word is always taken but a delay may occur between acceptance of first and second words. IOS 10116 is required to hold the second word valid on IOM Bus 10130 until MEM 10112 responds with TIOMD to indicate that the block operation may proceed.

Data Available for IO (DAVIO) is a signal asserted by MEM 10112 to IOS 10116 indicating that data requested by IOS 10116 is available. DAVIO is asserted by MEM 10112 during the system clock cycle in which MEM 10112 places the requested data on MIO Bus 10129. In any single word type transfer, DAVIO is active for a single system clock transfer. In block type transfers, DAVIO is normally active for four consecutive system clock cycles. Upon event of a single cycle "bubble" resulting from detection and

correction of an ERCC error by BC 20114, DAVIO will remain high for four non-consecutive system clock cycles and with a single cycle bubble, a non-assertion, in DAVIO corresponding to the detection and correction of the error.

IO Memory Interrupt (IMINT) is a signal asserted by MEM 10112 to IOS 10116 when BC 20114 places a record of a detected error in BC 20114's Error Log, as described above.

Previous MIO Transfer Invalid (PMIOI) signal is similarly a signal asserted by MEM 10112 to IOS 10116 regarding errors in data read from MEM 10112 to IOS 10116. If an uncorrectible error appears in such data, that is an error in two or more data bits, the incorrect data is read to IOS 10116 and PMIOI signal asserted by MEM 10112. Correctible, or single bit, errors in data do not result in assertion of PMIOI. MEM 10112 will assert PMIOI to IOS 10116 of the next system clock cycle following MEM 10112's assertion of DAVIO.

Having described MEM 10112's interface to IOS 10116, and certain operations which IOS 10116 may request of MEM 10112, certain MEM 10112 operations within the capability of the interface will be described next. First, operand transfers, for example of numeric data, between MEM 10112 and IOS 10116 may be bit granular with any length from one to sixteen bits. Operand transfers may cross boundaries within a page but may not cross physical page boundaries. As previously described, MIO Bus 10129 and IOM Bus 10130 are capable of transferring 32 bits of data at a time. The least significant 16 bits of these buses, that is bits 16 to 31, will contain right justified data during operand transfers. The contents of the most significant 16 bits of these buses is generally not defined as MEM 10112 generally does not perform fill operations on read operations to IO Port 20910, nor does IOS 10116 fill unused bits during write operations. During a read or write operation, only those data bits indicated by length field in the corresponding memory request are of significance. In all cases, however, parity must be valid on all 32 bits of MIO Bus 10129 and IOM Bus 10130.

Referring to Fig. 204, IOS 10116 includes Data Channels 20410 and 20412 each of which will be described further in a following detailed description of IOS 10116. Data Channels 20410 and 20412 each possess particular characteristics defining certain IO Port 20910 operations. Data Channel 20410 operates to read and write block aligned full and partial blocks. Full blocks have block aligned addresses and lengths of 16 bytes. Partial blocks have byte aligned addresses and lengths of 1 to 15 bytes; a partial block transfer must be within a block, that is not cross block boundaries. A full 4 word block will be transferred between IOS 10116 and MEM 10112 in either case, but only those blocks indicated by length of field in a corresponding MEM 10112 request are of actual significance in a write operation. Non-addressed bytes in such operations may contain any information so long as parity is valid for the entire data transfer. Data Channel 20412 preferably reads or writes 16 bits at a time on double byte boundaries. Such reads and writes are right justified on MIO Bus 10129 and IOM Bus 10130. The most significant 16 bits of these buses may contain any information during such operations so long as parity is valid for the entire 32 bits. Data Channel 20412 operations are similar to IOS 10116 operand read and write operations with double byte aligned addresses and lengths of 16 bits. Finally, instructions, for example controlling IOS 10116 operation, are read from MEM 10112 to IOS 10116 a block at a time. Such operations are identical to a full block data read.

Having described the operating characteristics of IO Port 20910, the operating characteristics of JPO Port 21010 will be described next.

2. JPO Port 21010 Operating Characteristics (Fig. 210)

Referring to Fig. 210, a diagramic representation of JPO Port 21010 is shown. As previously described, JPO Port 21010 is utilized for transfer of operands, for example numeric data, between MEM 10112 and JP 10114. JPO Port 21010 includes a request input (address, length, and operation information) to MIC 20122 from 36 bit PD Bus 10146, a write data input to FIU 20120 from 32 bit JPD Bus 10142, a 32 bit read data output from MC 20116 and FIU 20120 to 32 bit MOD Bus 10144, and bi-directional control inputs and outputs between MIC 20122 and JPMC Bus 10147.

Referring first to JPO Port 21010's read data output to MOD Bus 10144, MOD Bus 10144 is used by JPO Port 21010 to transfer data, for example operands, to JP 10114. MOD Bus 10144 is also utilized internal to MEM 10112 as a bidirectional bus to transfer data between MC 20116 and FIU 20120. In this manner, data may be transferred from MC 20116 to FIU 20120 where certain data format operations are performed on the data before the data is transferred to JP 10114 through MOD Bus 10144. Data may also be used to transfer data from FIU 20120 to MC 20116 after a data format operation is performed in a write operation. Data may also be transferred directly from MC 20116 to JP 10114 through MOD Bus 10144. Internal to MEM 10112, MOD Bus 10144 is a 36 bit bus for concurrent transfer of 32 bits of data, MOD Bus 10144 bits (MOD(0-31)), and 4 bits of odd parity, 1 bit per byte, MOD Bus 10144 bits (MODP(0-3)). External to MEM 10112, MOD Bus 10144 is a 32 bit bus, comprising bits (MOD(0-31)); parity bits are not read to JP 10114.

Data is written into MEM 10112 through JPD Bus 10142 to FIU 20120. As just described, data format operations may then be performed on this data before it is transferred from FIU 20120 to MC 20116 through MOD Bus 10144. In such operations, JPD Bus 10142 operates as a 32 bit bus carrying 32 bits of data, bits (JPD (0-31)), with no parity bits. JO Port 21010 generates parity for JPD Bus 10142 data to be written into MEM 10112 as this data is transferred into MEM 10112.

Memory requests are also transmitted to MEM 10112 from JP 10114 through JPD Bus 10142, which operates in this regard as a 40 bit bus. Each such request includes an address field, a length field, an FIU

EP 0 067 556 B1

field specifying data formatting operations to be performed, operation code field, and a destination code field specifying destination of data read from MEM 10112. Address field includes a 13 bit physical page number field, (JPPN(0—12)), and a 14 bit physical page offset field, (JPPO(0—13)). Length field includes 6 bits of length information, (JLNG(0—5)), and expresses true length of the data item to be written to or read from MEM 10112.

As JPD Bus 10142 and MOD Bus 10144 are each capable of transferring 32 bits of data in a single MEM 10112 read or write cycle, 6 bits of length information are required to express true length. As will be described in a following description, JP 10114 may provide physical page offset and length information directly to MEM 10112, performs logical page number to physical page number translations, and may perform a Protection Mechanism 10230 check on the resulting physical page number. As such, MEM 10112 expects to receive (JPPN(0—12)) later than (JPPO(0—13)) and (JLNG(0—5)). (JPPO(0—13)) and (JLNG(0—5)) should, however, be valid during the system clock cycle in which a JP 10114 memory request is loaded into MEM 10112.

Operation code field provided to MEM 10112 from JP 10114 is a 3 bit code, (JMCMD(0—2)) specifying an operation to be formed by MEM 10112. Certain operations which JP 10114 may request of MEM 10112, and their corresponding operation codes, are:

- 000 = read;
- 001 = read and set;
- 010 = write;
- 011 = error;
- 100 = error;
- 101 = read error log and reset;
- 110 = repair block; and,
- 111 = flush cache.

Two bit FIU field, (JFIU(0—1)) specifies data manipulation operations to be performed in executing read and write operations. Among the data manipulation operations which may be requested by JP 10114, and their FIU fields, are:

- 00 = right justified, zero fill;
- 01 = right justified, sign extend;
- 10 = left justify, zero fill; and,
- 11 = left justify, blank fill.

For write operations, JPO Port 21010 may respond only to the most significant bit of FIU field, that is the FIU field bit specifying alignment.

Finally, destination field is a two bit field specifying a JP 10114 destination for data read from MEM 10112. This field is ignored for write operations to MEM 10112. A first bit of destination field, JPMDST, identifies the destination to be FU 10120, and the second field, EBMDST, specifies EU 10120 as the destination.

JPMC Bus 10147 includes dedicated lines for exchange of control signals between JPO Port 21010 and JP 10114. Among these control signals is Load JO Request (LJOR), which is asserted by JP 10114 when JP 10114 wishes to load a request into MEM 10112. LJOR is asserted concurrently with presentation of the memory request to MEM 10112 through PD Bus 10146. JO Port Available (JOPA) is asserted by MEM 10112 when JPO Port 21010 is available to accept a new memory request from JP 10114. If LJOR and JOPA are asserted concurrently, MEM 10112 accepts the memory request from JP 10114 and MEM 10112 drops JOPA to indicate that memory request has been accepted. As previously discussed, MEM 10112 may assert JOPA while a previous request is being executed and the PD Bus 10146 information, that is the memory request previously provided concerning the previous request, is no longer required.

If JP 10114 submits a memory request and JOPA is not asserted by MEM 10112, MEM 10112 does not accept the request and JP 10114 must resubmit that request when JOPA is asserted. Because, as described above, JPPN field of a memory request from JP 10114 may arrive late compared to the other fields of the request, MEM 10112 will delay loading of JPPN field for a particular request until the next system clock cycle after the request was initially submitted. MEM 10112 may also obtain this JPPN field at the same time it is being loaded into the port register by by-passing the port register.

JP 10114 may abort a memory request upon asserting Abort JP Request (ABJR). ABJR will be accepted by MEM 10112 during system clock cycle after accepting memory request from JP 10114 and ABJR will result in cancellation of the requested operation. A single ABJR line is provided for both JPO Port 21010 and JPI Port 21110 because, as described in a following description, MEM 10112 may accept only a single request from JP 10114, to either JPO Port 21010 or to JPI port 21110, during a single system clock cycle.

Upon completion of an operand read operation requested through JPO Port 21010 MEM 10112 may assert either of two data available signals to JP 10114. These signals are data available for FA(DAVFA) and data available for EB(DAVEB). As previously described, a part of each read request from JP 10114 includes a destination field specifying the intended destination of the requested data. As will be described further

below, MEM 10112 tracks such destination information for read requests and returns destination information with a corresponding information in the form of DAVFA and DAVEB. DAVFA indicates a destination in FU 10120 while DAVEB indicates a destination in EU 10122. MEM 10112 may also assert signal zero filled (ZFILL) specifying whether read data for JPO Port 21010 is zero filled. ZFILL is valid only when DAVEB is asserted.

For JPO Port 21010 write request, the associated write data word should be valid on same system clock cycle as the request, or one system clock cycle later. JP 10114 asserts Load JP Write Data (LJWD) during the system clock cycle when JP 10114 places valid write data on JPD Bus 10142.

As previously discussed, when MEM 10112 detects an error in servicing a JP 10114 request MEM 10112 places a record of this error in MC 20116's Error Log. When an entry is placed in Error Log for either JPO Port 21010 or IO Port 20910, MEM 10112 asserts an interrupt flag signal indicating a valid Error Log entry is present. DP 10118 detects this flag signal and may direct the flag signal to either JP 10114 or IOS 10116, or both. IOS 10116 or JP 10114, as selected by DP 10118, may then read and reset Error Log and reset the flag. The interrupt flag signal is not necessarily directed to the requestor, JP 10114 or IOS 10116, whose request resulted in the error.

If an uncorrectable MEM 10112 error, that is an error in two or more bits of a single data word, is detected in a read operation the incorrect data is read to JP 10114 and an invalid data signal asserted. A signal, Previous MOD Transfer Invalid (PMDI), is asserted by MEM 10112 on the next system clock cycle following either DAVFA or DAVEB. PMDI is not asserted for single bit errors, instead the data is corrected and the corrected data read to JP 10114.

Having described JPO Port 21010's structure, and characteristics, JPI Port 21110 will be described next below.

3. JPI Port 21110 Operating Characteristics (Fig. 211)

Referring to Fig. 211, a diagramic representation of JPI Port 21110 is shown. JPI port 21110 includes an address input from PD Bus 10146 to FIU 20120, a data output to MOD Bus 10144 from MC 20116, and bi-directional control inputs and outputs from MIC 20122 to JPMC Bus 10147. As previously described, a primary function of JPI Port 21110 is the transfer of SOPs and operand NAMES from MEM 10112 to JP 10114 upon request from JP 10114. JPI Port thereby performs only read operations wherein each read operation is a transfer of a single 32 bit word having a word aligned address.

Referring to JPI Port 21110 input from PD Bus 10146, read requests to MEM 10112 by JP 10114 for SOPs and operand NAMES each comprise a 21 bit word address. As described above, each JPI Port 21110 read operation is of a single 32 bit word. As such, the five least significant bits of address are ignored by MEM 10112. For the same reason, a JPI Port 21110 request to MEM 10112 does not include a length field, an operation code field, an FIU field, or a destination code field. Length, operation code, and FIU code fields are not required since JPI Port 21110 performs only a single type of operation and destination code field is not required because destination is inherent in a JPI Port 21110 request.

The 32 bit words read from MEM 10112 in response to JPI Port 21110 requests are transferred to JP 10114 through MC 20116's 32 bit output to MOD Bus 10144. As in the case of JPO 21010 read outputs to JP 10114, JPI Port 21110 does not provide parity information to JP 10114.

Control signals exchange between JP 10114 and JPI Port 21110 through JPMC Bus 10147 include Load JI Request (LJIR) and JI Port Available (JIPA), which operate in the same manner as discussed with reference to JPO Port 21010. As previously described, JPO Port 21010 and JPI Port 21110 share a single Abort JP Request (ABJR) command. Similarly, JPO Port 21010 and JPI Port 21110 share previous MOD Transfer Invalid (PMDI) from MEM 10112. As described above, a JPI port 21110 request does not include a destination field as destination is implied. MEM 10112 does, however, provide a Data Available Signal (DAVFI) to JP 10114 when a word read from MEM 10112 in response to a JPI Port 21110 request is present on MOD Bus 10144 and valid.

Having described the overall structure and operation of MEM 10112, and the structure and operation of MEM 10112's interface to JP 10114 and IOS 10116, the structure and operation of FIU 20120 MEM 10112 will next be described in further detail.

h. FIU 20120 (Figs. 201, 230, 231)

As previously described, FIU 20120 performs certain data manipulation operations, including those operations necessary to make MEM 10112 bit addressable. Data manipulation operations may be performed on data being written into MEM 10112, for example, JP 10114 through JPD Bus 10142 or from IOS 10116 through IOM Bus 10130. Data manipulations operations may also be performed on data being read from MEM 10112 to JPD 10114 or IOS 10116. In case of data read to JP 10114, MOD Bus 10144 is used both as a MEM 10112 internal bus, in transferring data from MC 20116 to FIU 20120 for manipulation, and to transfer manipulated data from MEM 10112 to JP 10114. In case of data read to IOS 10116, MOD Bus 10144 is again used as MEM 10112 internal bus to read data from MC 20116 to FIU 20120 for subsequent manipulation. The manipulated data is then read from FIU 20120 to IOS 10116 through MIO Bus 10129.

Certain data manipulation operations which may be performed by FIU 20120 have been previously described. In general, a data manipulation operation consists of four distinct operations, and FIU 20120 may manipulate data in any possible manner which may be achieved through performing any combination

of these operations. These four possible operations are selection of data to be manipulated, rotation or shifting of that data, masking of that data, and transfer of that manipulated data to a selected destination. Each FIU 20120 data input will comprise a thirty-two bit data word and, as described above, may be selected from input provided from JPD Bus 10142, MOD Bus 10144, and IOM Bus 10130. In certain cases, an
 5 FIU 20120 data input may comprise two thirty-two bit words, for example, when a cross word operation is performed generating an output comprised of bits from each of two different thirty-two bit words. Rotation or shifting of a selected thirty-two bit data word enables bits within a selected word to be repositioned with respect to word boundaries. When used in conjunction with the masking operation, described momentarily, rotation and shifting may be reiterably performed to transfer any selected bits in a word to
 10 any selected locations in that word. As will be described further below, a masking operation allows any selected bits of a word to be affectively erased, thus leaving only certain other selected bits, or certain selected bits to be forced to predetermined values. A masking operation may be performed, for example, to zero fill or sign extend portions of a thirty-two bit word. In conjunction with a rotation or shifting operation, a masking operation may, for example, select a single bit of a thirty-two bit input word, position that bit in
 15 any selected bit location, and force all other bits of that word to zero. Each output of FIU 20120 is a thirty-two bit data word and, as described above, may be transferred on to MOD Bus 10144 or onto MIO Bus 10129. As will be described below, selection of a particular sequence of the above four operations to be performed on a particular data word is determined by control inputs provided from MIC 20122. These control inputs from MIC 20122 are decoded and executed by microinstruction control logic included within
 20 FIU 20120.

Referring to Fig. 230, a partial block diagram of FIU 20120 is shown. As indicated therein, FIU 20120 includes Data Manipulation Circuitry (DMC) 23010 and FIU Control Circuitry (FIUC) 23012. Data Manipulation Circuitry 23010 in turn includes FIUIO circuitry (FIUIO) 23014, Data Shifter (DS) 23016, Mask Logic (MSK) 23018, and Assembly Register (AR) logic 23020. Data manipulation circuitry 23010 will be
 25 described first followed by FIUC 23012. In describing data manipulation circuitry 23010, FIUIO 23014 will be described first, followed by DS 23016, MSK 23018, and AR 23020, in that order.

Referring to FIUIO 23014, FIUIO 23014 comprises FIU 20120's data input and output circuitry. Job Processor Write Data Register (JWDR) 23022, IO System Write Data Register (IWDR) 23024, and Write Input Data Register (RIDR) 23026 are connected from, respectively, JPD Bus 10142, IOM Bus 10130, and MOD Bus
 30 10144 for receiving data word inputs from, respectively, JP 10114, IOS 10116, and MC 20116. JWDR 23022, IWDR 23024 and RIDR 23026 are each thirty-six bit registers comprised, for example, of SN74S374 registers. Data words transferred into IWDR 23024 and RIDR 23026 are each, as previously described, comprised of a thirty-two data word plus four bits of parity. Data inputs from JP 10114 are, however, as previously described, thirty-two bit data words without parity. Job Processor Parity Generator (JPPG)
 35 23028 associated with JWDR 23022 is connected from JPD Bus 10142 and generates four bits of parity for each data input to JWDR 23022. JWDR 23022's thirty-six bit input thereby comprises thirty-two bits of data, directly from JPD Bus 10142, plus a corresponding four bits of parity from JPPG 23028.

Data words, thirty-two bits of data plus four bits of parity, are transferred into JWDR 23022, IWDR 23024, or RIDR 23026 when, respectively, input enable signals Load JWD (LJWD), Load IWD (LIWD) or Load RID (LRID) are asserted. LJWD is provided from FU 10120 while LIWD and LRID are provided from MIC
 40 20122.

Data words resident in JWDR 23022, IWDR 23024, or RIDR 23026 may be selected and transferred onto FIU 20120's Internal Data (IB) Bus 23030 by output enable signals JWD Enable Output (JWDEO), IWD Enable Output (IWDEO), an RID Enable Output (RIDEO). JWDEO, IWDEO, and RIDEO are provided from
 45 FIUC 23012 described below.

As will be described further below, manipulated data words from DS 23016 or AR 23020 will be transferred onto, respectively, Data Shifter Output (DSO) Bus 23032 or Assembly Register Output (ASYRO) Bus 23034 for subsequent transfer onto MOD Bus 10144 or MIO Bus 10129. Each manipulated data word appearing on DSO Bus 23032 or ASYRO Bus 23034 will be comprised of 32 bits of data plus 4 bits of parity.
 50 Manipulated data words present on DSO Bus 23032 may be transferred onto MOD Bus 10144 or MIO Bus 10129 through, respectively, DSO Bus To MOD Bus Driver Gate (DSMOD) 23036 or BSO Bus To MIO Bus Driver Gate (DSMIO) 23038. Manipulated data words present on ASYRO Bus 23034 may be transferred onto MOD Bus 10144 or MIO Bus 10129 through, respectively, ASYRO Bus To MOD Bus Driver Gate (ASYMOD) 23040 or ASYRO Bus To MIO Bus Driver Gate (ASYMIO) 23042. DSMOD 23036, DSMIO 23038, ASYMOD 23040, and ASYMIO 23042 are each comprised of, for example, SN74S244 drivers. A manipulated data
 55 word on DSO Bus 23032 be transferred through DSMOD 23036 to MOD Bus 10144 when driver gate enable signal Driver Shift To MOD (DRVSHFMOD) to DSMOD 23036 is asserted. Similarly, a manipulated data word on DSO Bus 23032 will be transferred through DSMIO 23038 to MIO Bus 10129 when driver gate enable signal Drive Shift Through MIO Bus (DRVSHFMIO) to DSMIO 23038 is asserted. Manipulated data
 60 words present on ASYRO Bus 23034 may be transferred onto MOD Bus 10144 or MIO Bus 10129 when, respectively, driver gate enable signal Drive Assembly To Mod Bus (DRVASYMOD) to ASYMOD 23040 or Drive Assembly To MIO Bus (DRVASYMIO) to ASYMIO 23042 are asserted. DRVSHFMOD, DRVSHFMIO, DRVASYMOD, and DRVASYMIO are provided, as described below, from FIUC 23012.

Registers IARM 23044 and BARMR 23046, which will be described further in a following description of
 65 DP 10118, are used by DP 10118 to, respectively, write data words onto IB 23030 and to Read data words

from MOD Bus 10144, for example manipulated data words from FIU 20120. Data word written into IARMR 23044 from DP 10118, that is 32 bits of data and 4 bits of parity, will be transferred onto IB Bus 23030 when register enable output signal IARM enable output (IARMEO) from FIUC 23012 is asserted. Similarly, a data word present on MOD Bus 10144, comprising 32 bits of data plus 4 bits of parity, will be written into BARMR 23046 when load enable signal Load BARMR (LDBARMR) to BARMR 23046 is asserted by MIC 20122. A data word written into BARMR 23046 from MOD Bus 10144 may then subsequently be read to DP 10118. IARMR 23044 and BARMR 23046 are similar to JWDR 23022, IWDR 23024, and IRDR 23026 and may be comprised, for example, of SN74S299 registers.

Referring finally to IO Parity Check Circuit (IOPC) 23048, IOPC 23048 is connected from IB Bus 23030 to receive each data word, that is 32 bits of data plus 4 bits of parity, appearing on IB Bus 23030. IOPC 23048 confirms parity and data validity of each data word appearing on IB Bus 23030 and, in particular, determines validity of parity and data of data words written into FIU 20120 from IOS 10116. IOPC 23048 generates output Parity Error (PER), previously discussed, indicating a parity error in data words from IOS 10116.

Referring to DS 23016, DS 23016 includes Byte Nibble Logic (BYNL) 23050, Parity Rotation Logic (PRL) 23052, and Bit Scale Logic (BSL) 23054. BYNL 23050, PRL 23052, and BSL 23054 may respectively be comprised of, for example, 25S10 shifters. BYNL 23050 is connected from IB Bus 23030 for receiving and shifting the 32 data bits of a data word selected and transferred onto IB Bus 23030. PRL 23052 is a 4 bit register similarly connected from IB Bus 23030 to receive and shift the 4 parity bits of a data word selected and transferred onto IB Bus 23030. Outputs of BYNL 23050 and PRL 23052 are both connected onto DSO Bus 23032, thus providing a 36 bit FIU 20120 data word output directly from BYNL 23050 and PRL 23052. BYNL 23050's 32 bit data output is also connected to BSL 23054's input. BSL 23054's 32 bit output is in turn provided to MSK 23018.

As previously described, DS 23016 performs data manipulation operations involving shifting of bits within a data word. In general, data shift operations performed by DS 23016 are rotations wherein data bits are right shifted, with least significant bits of data word being shifted into most significant bit position and most significant bits being translated towards least significant bit positions. DS 23016 rotation operations are performed in two stages. First stage is performed by BYNL 23050 and PRL 23052 and comprises right rotations on a nibble basis (a nibble is defined as 4 bits of data). That is, BYNL 23050 right shifts a data word by an integral number of 4 bit increments. A right rotation on a nibble by nibble basis may, for example, be performed when RM 20722 asserts FLIPHALF previously described. FLIPHALF is asserted for IOS 10116 half word read operations wherein the request data resides in the most significant 16 bits of a data word from MC 20116. BYNL 23050 will perform a right rotation of 4 nibbles to transfer the desired 16 bits of data into the least significant 16 bits of BYNL 23050's output. Resulting BYNL 23050 output, together with PRL 23052's parity bit output would then be transferred through DSO 23032 to MIO Bus 10129. In addition to performing data shifting operations, DS 23016 may transfer a data word, that is the 32 bits of data, directly to MSK 23018 when data manipulation to be performed does not require data shifting, that is shifts of 0 bits may be performed.

Because data bits are shifted by BYNL 23050 on a nibble basis, the relationship between the 32 data bits of a word and the corresponding 4 parity bits may be maintained if parity bits are similarly right rotated by an amount corresponding to right rotation of data bits. This relationship is true if the data word is shifted in multiples of 2 nibbles, that is 8 bits or 1 byte. PRL 23052 right rotates the 4 parity bits of a data word by an amount corresponding to right rotation of the corresponding 32 data bits in BYNL 23050. Right rotated outputs of BYNL 23050 and PRL 23052 therefore comprise a valid data word having 32 bits of data and 4 bits of parity wherein the parity bits are correctly related to the data bits. A right rotated data word output from BYNL 23050 and PRL 23052 may be transferred onto DSO Bus 23032 for subsequent transfer to MOD Bus 10144 or MIO Bus 10129 as described above. DSO 23032 is used as FIU 20120's output data path for byte write operations and "rotate read" operations wherein the required manipulation of a particular data word requires only an integral number of right rotations by bytes. Amount of right rotation of 32 bits of data in BYNL 23050 and 4 bits of parity in PRL 23052 is controlled by input signal shift (SHFT) (0—2) to BYNL 23050 and PRL 23052. As will be described below, SHFT (0—2) is generated, together with SHFT (3—4) controlling BSL 23054, by FIUC 23012. BYNL 23050 and PRL 23052, like BSL 23054 described below, are parallel shift logic chips and entire rotation operation of BYNL 23050 and PRL 23052 or BSL 23054 may be performed in a single clock cycle.

Second stage of rotation is performed by BSL 23054 which, as described above, receives the 32 data bits of a data word from BYNL 23050. BSL 23054 performs right rotation on a bit by bit basis with the shift amount being selectable between 0—3 bits. Therefore, BSL 23054 may rotate bits through nibble boundaries. BYNL 23050 and BSL 23054 therefore comprise a data shifting circuit capable of performing bit-by-bit right rotation by an amount from 1 bit to a full 32 bit right rotation.

Referring now to MSK 23018, MSK 23018 is comprised of 5 32 bit Mask Word Generators (MWG's) 23056 to 23064. MSK 23018 generates a 32 bit output to AR 23020 by selectively combining 32 bit mask word outputs of MWG's 23056 to 23064. Each mask word generated by one of MWG's 23056 to 23064 is effectively comprised a bit by bit combination of a set of enabling bits and a predetermined 32 bit mask word, generated by FIUC 23012 and MIC 20122. MWG's 23058 to 23064 are each comprised of for example, open collector NAND gates for performing these functions, while MWG 23056 is comprised of a PROM.

As just described, outputs of MWG's 23056 to 23064 are all open collector circuits so that any selected combination of mask word outputs from MWG's 23056 to 23064 may be ORed together to comprise the 32 bit output of MSK 23018.

MWG 23058 to MWG 23064 generate, respectively, mask word outputs Locked Bit Word (LBW) (0—31), Sign Extended Word (SEW) (0—31), Data Mask Word (DMW) (0—31), Blank Fill Word (BWF) (0—31), and Assembly Register Output (ARO) (0—31). Referring first to MWG 23064 and ARO (0—31), the contents of Assembly Register (ASYMR) 23066 in AR 23020 are passed through MWG 23064 upon assertion of enabling signal Assembly Output Register (ASYMOR). ARO (0—31) is thereby a copy of the contents of ASYMR 23066 and MWG 23064 allows the contents of ASYMR 23066 to be ORed with the selected combination of LBW (0—31), SEW (0—31), DMW (0—31), or BFW (0—31).

DMW (0—31) from MWG 23060 is generated by ANDing enable Input Data Mask (DMSK) (0—31) with the 32 bit output of DS 23016. DMSK (0—31) is a 32 bit enabling word generated, as described below, by FIUC 23012. FIUC 23012 may generate 4 different DMSK (0—31) patterns. Referring to Fig. 231, the 4 DMSKs (0—31) which may be generated by FIUC 20132 are shown. DMSKA (0—31) is shown in Line A of Fig. 231. In DMSKA (0—31) all bits to the left of but not including a bit designated by Left Bit Address (LBA) and all bits to the right of and not including a bit designated by Right Bit Address (RBA) are 0. All bits between, and including, those bits designated by LBA and RBA are 1's. DMSKB (0—31) is shown in Line B of Fig. 231 and is DMSKA (0—31) inverted. DMSKC (0—31) and DMSKD (0—31) are shown, respectively, in Lines C and D of Fig. 231 and are comprised of, respectively, all 0's or all 1's. As stated above DMSK (0—31) is ANDed with the 32 bit output of DS 23016. As such, DMSKC (0—31) may be used, for example, to inhibit DS 23016's output while DMSKD (0—31) may be used, for example, to pass DS 23016's output to AR 23020. DMSKA (0—31) and DMSKB (0—31) may be used, for example, to gate selected portions of DS 23016's output to AR 23020 where, for example, the selected portions of DS 23016's output may be ORed with other mask word outputs MSK 23018.

Referring next to MWG 23062, MWG 23062 generates BFW (0—31). BFW (0—31) is used in a particular operation wherein 32 bit data words containing 1 to 4 ASCII blanks are required to be generated wherein 1 bit/byte contains a logic one and remaining bits contain logic zeros. In this case, the ASCII blank bytes may contain logic 1's in bit positions 2, 10, 18, and 26.

Referring again to Fig. 231, Line E therein shows 32 bit right mask (RMSK) (0—31) which may be generated by FIUC 23012. In the most general case, RMSK contains zeros in all bit positions to the left of and including a bit position designated by RBA. When used in a blank fill operation, bit positions 2, 10, 18, and 26 may be selected to contain logic 1's depending upon those byte positions containing logic 1's, that is in those bytes containing ASCII blanks; these bytes to the right of RBA are determined by RMSK (0—31). RMSK (0—31) is enabled through MWG 23062 as BWF (0—31) when MWG 23062 is enabled by blank fill (BLNKFILL) provided from FIU 23012.

As described above, MWG's 23058 to 23064 and in particular MWG's 23060 and MWG 23062 are NAND gate operations. Therefore, the outputs of MWGs 23056 through 23064 are active low signals. The inverted output of ASYMR 23066 is used as an output to ASYRO 23034 to invert these outputs to active high.

MWG 23058, generating SEW (0—31), is used in generating sign extended or filled words. In sign extended words, all bit spaces to the left of the most significant bit of a 32 bit data word are filled with the sign bit of the data contained therein, the left most bits of the 32 bit word are filled with 1's or 0's depending on whether that word's sign bit indicates that the data contained therein is a positive or negative number.

Sign Select Multiplexor (SIGNSEL) 23066 is connected to receive the 32 data bits of a word present on IB Bus 23030. Sign Select (SGNSEL) (0—4) to SIGNSEL 23066 is derived from SBA (0—4), that is from SBA Bus 21226 from PRMUX 20720. As previously described, SBA (0—4) is Starting Bit Address identifying the first or most significant bit of a data word. When a data word contains a signed number, most significant bit contains sign bit of that number. SGNSEL (0—4) input to SIGNSEL 23066 is used as a selection input and, when SIGNSEL is enabled by Sign Extend (SIGNEXT) from FIU 23012, selects the sign bit on IB Bus 23030 and provides that sign bit as an input to MWG 23058.

Sign bit input to MWG 23058 is ANDed with each bit of left hand mask (LMSK) (0—31) from FIUC 23012. Referring again to Fig. 231, LMSK (0—31) is shown on Line F thereof. LMSK (0—31) contains all 0's to the right of and including the bit space identified by LBA and 1's in all bit spaces to the left of that bit space identified by LBA. SEW (0—31) will therefore contain sign bit in all bit spaces to the left of the most significant bit of the data word present on output of MWG 23058. The data word on IB Bus 23030 may then be passed through DS 23016 and subjected to a DMSK operation wherein all bits to the left of the most significant bit are forced to 0. SEW (0—31) and DMW (0—31) outputs of MWG's 23058 and 23060 may then be ORed to provide the desired sign extended word output.

LBW (0—31), provided by MWG 23056, is used in locked bit operations wherein the most significant data bit of a data word is in MEM 10112 forced to logic 1. SIGNSEL (0—4) is an address input to MWG 23056 and, as previously described, indicates most significant data bit of a data word present on an IB Bus 23030. MWG 23056 is enabled by input Lock (LOCK) from FIUC 23012 and the resulting LBW (0—31) will contain a single logic 1 in the bit space of the most significant data bit of the data word present on IB Bus 23030. The data word present on IB Bus 23030 may then be passed through DS 23016 and MWG 23060 to be ORed with LBW (0—31) so that that data words most significant data bit is forced to logic 1.

Referring to AR 23020, AR 23020 includes ASYMR 23066, which may be comprised for example of a

SN74S175 registers, and Assembly Register Parity Generator (ASYPG) 23070. As previously described, ASYMR 23066 is connected from MSK 23018 32 bit output. A 32 bit word present on MSK 23018's output will be transferred into ASYMR 23066 when ASYMR 23066 is enabled by Assembly Register Load (ASYMLD) from MIC 20122. The 32 bit word generated through DS 23016 and MSK 23018 will then be present on ASYRO Bus 23034 and may, as described above, then be transferred onto MOD Bus 10144 or MIO Bus 10129. ASYPG 23070 is connected from ASYMR 23066 32 bit output and will generate 4 parity bits for the 32 bit word presently on the 32 data lines of ASYRO Bus 23034. ASYPG 23070's 4 bit parity output is based on the 4 parity bit lines of ASYRO Bus 23034 and accompany the 32 bit data word present thereon.

Having described structure and operation of Data Manipulation Circuitry 23010, FIUC 23012 will be described next below.

Referring again to Fig. 230, FIUC 23012 provides pipelined microinstruction control of FIU 20120. That is, control signals are received from MIC 20122 during a first clock cycle and certain of the control signals are decoded by microinstruction logic to generate further FIUC 23012 control signals. During the second clock cycle, control signals received and generated during first clock cycle are provided to DMC 23010, some of which are further decoded to provide yet other control signals to control operation of FIUC 23012. FIUC 23012 includes Initial Decode Logic (IDL) 23074, Pipeline Registers (PPLR) 23072, Final Decoding Logic (FDL) 23076, and Enable Signal Pipeline Register (ESPR) 23098 with Enable Signal Decode Logic (ESDL) 23099.

IDL 23074 and Control Pipeline Register (CPR) 23084 of PPLR 23072 are connected from control outputs of MIC 20122 to receive control signals therefrom during a first clock cycle as described above. IDL 23074 provides outputs to control pipeline registers Right Bit Address Register (RBAR) 23086, Left Bit Address Register (LBAR) 23088 and Shift Register (SHFR) 23090 of PPLR 23072. CPR 23084 and SHFR 23090 provide control outputs directly to DMC 23010. As described above these outputs control DMC 23010 during second clock cycle.

CPR 23084, RBAR 23086, and LBAR 23088 provide outputs to FDL 23076 during second clock cycle and FDL 23076 in turn provides certain outputs directly to DMC 23010.

ESPR 23098 and ESDL 23099 receive enable and control signals from MIC 20122 and in turn provide enable and control signals to DMC 23010 and certain other portions of MEM 10112 circuitry.

IDL 23074 and FDL 23076 may be comprised, for example, of PROMs. CPR 23084, RBAR 23086, LBAR 23088, SHFR 23090, and ESPR 23098 may be comprised, for example, of SN74S194 registers. ESDL 23099 may be comprised of, for example, compatible decoders, such as logic gates.

Referring first to IDL 23074, IDL 23074 performs an initial decoding of circuitry control signals from MIC 20122 and provides further control signals used by FIUC 23012 in controlling FIU 20120. IDL 23074 is comprised of read-only memory arrays Right Bit Address Decoding Logic (RBADL) 23078, Left Bit Address Decoding Logic (LBADL) 23080, and Shift Amount Decoding Logic (SHFAMTDL) 23082. RBADL 23078 receives, as address inputs, Final Bit Address (FBA) (0-4), Bit Length Number (BLN) (0-4), and Starting Bit Address (SBA) (0-4). FBA, BLN and SBA define, respectively, the final bit, length, and starting bit of a requested data item as previously discussed with reference to PRMUX 20720. RBADL 23078 also receives chip select enable signals Address Translation Chip Select (ATCS) 00, 01, 02, 03, 04, and 15 from MIC 20122 and, in particular, RM 20722. When FIU 20120 is required to execute certain MSK 23018 operations, inputs FBA (0-4), BLN (0-4), and SBA (0-4), together with an ATCS input, are provided to RBADL 23078 from MIC 20122. RBADL 23078 in turn provides output RBA (Right Bit Address) (0-4), which has been described above with reference to DMSK (0-31) and RMSK (0-31). LBADL 23080 is similar to RBADL 23078 and is provided with inputs BLN (0-4), FBA (0-4), SBA (0-4), and ATCS 06, 07, 08, 09, and 05 from MIC 20122. Again, for certain MSK 23018 operations, LBADL 23080 will generate Left Bit Address (LBA) (0-4), which has been previously discussed above with reference to DMSK (0-31) and LMSK (0-31).

RBA (0-4) and LBA (0-4) are, respectively, transferred to RBAR 23086 and LBAR 23088 at start of second clock cycle by Pipeline Load Enable signal PIPELD provided from MIC 20122. RBAR 23086 and LBAR 23088 in turn respectively provide outputs Register Right Address (RRAD) (0-4) and Register Left Address (RLAD) (0-4) as address inputs to Right Mask Decode Logic (RMSKDL) 23092, Left Mask Decode Logic (LMSKDL) 23094, and FDL 23076 at start of second clock cycle. RRAD (0-4) and RLAD (0-4) correspond respectively to RBA (0-4) and LBA (0-4).

RMSKDL 23092 and LMSKDL 23094 are ROM arrays, having, as just described, RRAD (0-4) and RLAD (0-4) as, respectively, address inputs and Mask Enable (MSKENBL) from CPR 23084 as enable inputs. Together, RMSKDL 23092 and LMSKDL 23094 generate, respectively, RMSK (0-31) and LMSK (0-31) to MSK 23018. RMSK (0-31) and LMSK (0-31) are provided as inputs to Exclusive Or/Exclusive Nor gating (XOR/XNOR) 23096. XOR/XNOR 23096 also receives enable and selection signal Out Mask (OUTMSK) from CPR 23084. RMSK (0-31) and LMSK (0-31) inputs to XOR/XNOR 23096 are used, as selected by OUTMSK from CPR 23084, to generate a selected DMSK (0-31) as shown in Fig. 231. DMSK (0-31) output of XOR/XNOR 23096 is provided, as described above, to MSK 23018.

Referring again to IDL 23074, SHFAMTDL 23082 decodes certain control inputs from MIC 20122 to generate, through SHFR 23090, control inputs SHFT (0-4) and SGNSEL (0-4) to, respectively, DS 23016, SGNSEL 23068 and MWG 23056. Address inputs to the PROMs comprising SHFAMTDL 23082 include FBA (0-4), SBA (0-4), and FLIPHALF (FLIPHALF) from MIC 20122. FBA (0-4) and SBA (0-4) have been described above. FLIPHALF is a control signal indicating that, as described above, that 16 bits of data

requested by IOS 10116 resides in the upper half of a 32 bit data word and causes those 16 bits to be transferred to the lower half of FIU 20120's output data word onto MIO Bus 10129. MIC 20122 also provides chip enable signals ATCS 10, 11, 12, 13, and 14. Upon receiving these control inputs from MIC 20122, SHFAMTDL 23082 generates an output shift amount (SHFAMT) (0—4) which, together with SBA (0—4) from MIC 20122, is transferred into SHFR 23090 by PIPELD at start of second clock cycle. SHFR 23090 then provides corresponding outputs SHFT (0—4) and SIGNSEL (0—4). As described above, SIGNSEL (0—4) are provided to SIGNSEL 23068 and MWG 23056 and MSK 23018. SHFT (0—4) is provided as SHFT (0—2) and SHFT (3—4) to, respectively, BYNL 23050 and BSL 23054 and DS 23016.

Referring to CPR 23084, as described above certain control signals are provided directly to FIU 20120 circuitry without being decoded by IDL 23074 or FDL 23076. Inputs to CPR 23084 include Sign Extension (SIGNEXT) and Lock (LOCK) indicating, respectively, that FIU 20120 is to perform a sign extension operation through MWG 23058 or a lock bit word operation through MWG 23056. CPR 23084 provides corresponding outputs SIGNEXT and LOCK to MSK 23018 to select these operations. Input Assembly Output Register (ASYMOR) and Blank Fill (BLANKFILL) are passed through CPR 23084 as ASYMOR and BLANKFILL to, respectively, MWG 23064 and MWG 23062 to select the output of ASYMR 23066 as a mask or to indicate that MSK 23018 is to generate a blank filled word through MWG 23062. Inputs OUTMSK and MSKENBL to CPR 23084 are provided, as discussed above, as enable signals OUTMSK and MSKENBL to, respectively, EXOR/ENOR 23096 and RMSKDL 23092 and LMSKBL 23094 and generating RMSK (0—31), LMSK (0—31), and DMSK (0—31) as described above.

Referring finally to ESPR 23098 and ESDL 23099, ESPR 23098 and PPLR 23072 together comprise a pipeline register and ESDL 23099 decoding logic for providing enable signals to FIU 20120 and other MEM 10112 circuitry. ESPR 23098 receives inputs Drive MOD Bus (DRVMOD) (0—1), Drive MIO Bus (DRVMIO) (0—1), and Enable Register (ENREG) (0—1) from MIC 20122 as previously described. DRVMOD (0—1), DRVMIO (0—1), and ENREG (0—1) are transferred into ESPR 23098 by PIPELD as previously described with reference to PPLR 23072. ESPR 23098 provides corresponding outputs to ESDL 23099, which in turn decodes DRVMOD (0—1), DRVMIO (0—1), and ENREG (0—1) to provide enable signals to FIU 20120 and other MEM 10112 circuitry. Outputs DRVSHFMOD, DRVASYMOD, DRVSHFMIO, and DRVASYMIO are provided to DSMOD 23036, DSMIO 23038, ASYMOD 23040, ASYMIO 23042, and FIUIO 23014 to control transfer of FIU 20120 manipulated data words onto MOD Bus 10144 and MIO Bus 10129. Outputs IARMEO, JWDEO, IWDEO, and RIDEO are provided as output enable signals to IARMR 23044, JWDR 23022, IWDR 23024, and RIDR 23026 to transfer the contents of these registers onto IB Bus 23030 as previously described. Outputs DRVCAMOD, DRVAMIO, DRVBYMOD, and DRVBYMIO are provided to MC 20116 for use in controlling transfer of information onto MOD Bus 10144 and MIO Bus 10129.

Having described the structure and operation of MEM 10112 above, the structure and operation of FU 10120 will be described next below.

B. Fetch Unit 10120 (Figs. 202, 206, 101, 103, 104, 238)

As has been previously described, FU 10120 is an independently operating, microcode controlled machine comprising, together with EU 10122, CS 10110's micromachine for executing user's programs. Principal functions of FU 10120 include: (1) Fetching and interpreting instructions, that is SInS comprising SOPs and Names, and data from MEM 10112 for use by FU 10120 and EU 10122; (2) Organizing and controlling flow and execution of user programs; (3) Initiating EU 10122 operations; (4) Performing arithmetic and logic operations on data; (5) Controlling transfer of data from FU 10120 and EU 10122 to MEM 10112; and, (6) Maintaining certain stack register mechanisms. Among these stack and register mechanisms are Name Cache (NC) 10226, Address Translation Cache (ATC) 10228, Protection Cache (PC) 10234, Architectural Base Registers (ABRs) 10364, Micro-Control Registers (mCRs) 10366, Micro-Stack (MIS) 10368, Monitor Stack (MOS) 10370 of General Register File (GRF) 10354, Micro-Stack Pointer Register Mechanism (MISPR) 10356, and Return Control Word Stack (RCWS) 10358. In addition to maintaining these FU 10120 resident stack and register mechanisms, FU 10120 generates and maintains, in whole or part, certain MEM 10112 resident data structures. Among these MEM 10112 resident data structures are Memory Hash Table (MHT) 10716 and Memory Frame Table (MFT) 10718, Working Set Matrix (WSM) 10720, Virtual Memory Management Request Queue (VMMRQ) 10721, Active Object Table (AOT) 10712, Active Subject Table (AST) 10914, and Virtual processor State Blocks (VPSBs) 10218. In addition, a primary function of FU 10120 is the generation and manipulation of logical descriptors which, as previously described, are the basis of CS 10110's internal addressing structure. As will be described further below, while FU 10120's internal structure and operation allows FU 10120 to execute arithmetic and logic operations, FU 10120's structure includes certain features to expedite generation and manipulation of logical descriptors.

Referring to Fig. 202, a partial block diagram of FU 10120 is shown. To enhance clarity of presentation, certain interconnections within FU 10120, and between FU 10120 and EU 10122 and MEM 10112 are not shown by line connections but, as described further below, are otherwise indicated, such as by common signal names. Major functional elements of FU 10120 include Descriptor Processor (DESP) 20210, MEM 10112 Interface Logic (MEMINT) 20212, and Fetch Unit Control Logic (FUCTL) 20214. DSP 20210 is, in general, an arithmetic and logic unit for generating and manipulating entries for MEM 10112 and FU 10120 resident stack mechanisms and caches, as described above, and, in particular, for generation and manipulation of logical descriptors. In addition, as stated above, DSP 20210 is a general purpose Central

Processor Unit (CPU) capable of performing certain arithmetic and logic functions.

DESP 20210 includes AON Processor (AONP) 20216, Offset Processor (OFFP) 20218, Length Processor (LENP) 20220. OFFP 20218 comprises a general, 32 bit CPU with additional structure to optimize generation and manipulation of offset fields of logical descriptors. AONP 20216 and LENP 20220 comprise, respectively, processors for generation and manipulation of AON and length fields of logical descriptors and may be used in conjunction with OFFP 20218 for execution of certain arithmetic and logical operations. DESP 20210 includes GRF 10354, which in turn include Global Registers (GRs) 10360 and Stack Registers (SRs) 10362. As previously described, GR's 10360 includes ABRs 10364 and mCRs 10366 while SRs 10362 includes MIS 10368 and MOS 10370.

MEMINT 20212 comprises FU 10120's interface to MEM 10112 for providing Physical Descriptors (physical addresses) to MEM 10112 to read SInS and data from and write data to MEM 10112. MEMINT 20212 includes, among other logic circuitry, MC 10226, ATC 10228, and PC 10234.

FUCTL 20214 controls fetching of SInS and data from MEM 10112 and provides sequences of microinstructions for control of FU 10120 and EU 10122 in response to SOPs. FUCTL 20214 provides Name inputs to MC 10226 for subsequent fetching of corresponding data from MEM 10112. FUCTL 20214 includes, in part, MISPR 10356, RCWS 10358, Fetch Unit S-Interpreter Dispatch Table (FUSDT) 11010, and Fetch Unit S-Interpreter Table (FUSITT) 11012.

Having described the overall structure of FU 10120, in particular with regard to previous descriptions in Chapter 1 of this description, DESP 20210, MEMINT 20212, and FUCTL 20214 will be described in further detail below, and in that order.

1. Description Processor 20210 (Figs. 202, 101, 103, 104, 238, 239)

As described above, DESP 20210 comprises a 32 bit CPU for performing all usual arithmetic and logic operations on data. In addition, a primary function of DESP 20210 is generation and manipulation of entries for, for example, Name Tables (NTs) 10350, ATC 10228, and PC 10234, and generation and manipulation of logical descriptors. As previously described, with reference to CS 10110 addressing structure, logical descriptors are logical addresses, or pointers, to data stored in MEM 10112. Logical descriptors are used, for example, as architectural base pointers or microcontrol pointers in ABRs 10364 and mCRs 10366 as shown in Fig. 103, or as linkage and local pointers of Procedure Frames 10412 as shown in Fig. 104. In a further example, logical descriptors generated by DESP 20210 and corresponding to certain operand Names are stored in MC 10226, where they are subsequently accessed by those Names appearing in SInS fetched from MEM 10112 to provide rapid translation between operand Names and corresponding logical descriptors.

As has been previously discussed with reference to CS 10110 addressing structure, logical descriptors provided to ATU 10228, from DESP 20210 or NC 10226, are translated by ATU 10228 to physical descriptors which are actual physical addresses of corresponding data stored in MEM 10112. That data subsequently is provided to JP 10114, and in particular to FU 10120 or EU 10122, through MOD Bus 10144.

As has been previously discussed with reference to MEM 10112, each data read to JP 10114 from MEM 10112 may contain up to 32 bits of information. If a particular data item referenced by a logical descriptor contains more than 32 bits of data, DESP 20210 will, as described further below, generate successive logical descriptors, each logical descriptor referring to 32 bits or less of information, until the entire data item has been read from MEM 10112. In this regard, it should be noted that NC 10226 may contain logical descriptors only for data items of 255 bits or less in length. All requests to MEM 10112 for data items greater than 32 bits in length are generated by DESP 20210. Most of data items operated on by CS 10110 will, however, be 32 bits or less in length so that NC 10226 is capable of handling most operand Names to logical descriptor translations.

As described above, DESP 20210 includes AONP 20216, OFFP 20218, and LENP 20220. OFFP 20218 comprises a general purpose 32 bit CPU with additional logic circuitry for generating and manipulating table and cache entries, as described above, and for generating and manipulating offset fields of AON pointers and logical descriptors. AONP 20216 and LENP 20220 comprise logic circuitry for generating and manipulating, respectively, AON and length fields of AON pointers and logical descriptors. As indicated in Fig. 202, GRF 10354 is vertically divided in three parts. A first part resides in ANOP 20216 and, in addition to random data, contains AON fields of logical descriptors. Second and third parts reside, respectively, in OFFP 20218 and LENP 20220 and, in addition to containing random data, respectively contain offset and length fields of logical descriptors. AON, Offset, and length portions of GRF 10354 residing respectively in AONP 20216, OFFP 20218, and LENP 20220 are designated, respectively, as AONGRF, OFFGRF, and LENGRF. AONGRF portion of GRF 10354 is 28 bits wide while OFFGRF and LENGRF portions of GRF 10354 are 32 bits in width. Although shown as divided vertically into three parts, GRF 10354 is addressed and operates as a unitary structure. That is, a particular address provided to GRF 10354 will address corresponding horizontal segments of each of GRF 10354's three sections residing in AONP 20216, OFFP 20218, and LENP 20220.

a. Offset Processor 20218 Structure

Referring first to OFFP 20218, in addition to being a 32 bit CPU and generating and manipulating table and cache entries and offset fields of AON pointers and logical descriptors, OFFP 20218 is DESP 20210's

primary path for receiving data from and transferring data to MEM 10112. OFFP 20218 includes Offset Input Select Multiplexer (OFFSEL) 20238, OFFGRF 20234, Offset Multiplexer Logic (OFFMUX) 20240, Offset ALU (OFFALU) 20242, and Offset ALU A Inputs Multiplexer (OFFALUSA) 20244.

5 OFFSEL 20238 has first and second 32 bit data inputs connected from, respectively, MOD Bus 10144 and JPD Bus 10142. OFFSEL 20238 has a third 32 bit data input connected from a first output of OFFALU 20242, a fourth 28 bit data input connected from a first output of AONGRF 20232, and a fifth 32 bit data input connected from OFFSET Bus 20228. OFFSEL 20238 has a first 32 bit output connected to input of OFFGRF 20234 and a second 32 bit output connected to a first input of OFFMUX 20240. OFFMUX 20240 has second and third 32 bit data inputs connected from, respectively, MOD Bus 10144 and JPD Bus 10142. OFFMUX 10 20240 also has a fourth 5 bit data input connected from Bias Logic (BIAS) 20246 and LENP 20220, described further below, and fifth 16 bit data input connected from NAME Bus 20224. Thirty-two bit data output of OFFGRF 20234 and first 32 bit data output of OFFMUX 20240 are connected to, respectively, first and second data inputs of OFFALUSA 20244. A first 32 bit data output of OFFALUSA 20244 and a second 32 bit data output of OFFMUX 20240 are connected, respectively, to first and second data inputs of OFFALU 15 20242. A second 32 bit data output of OFFALUSA 20244 is connected to OFFSET Bus 20228. A first 32 bit data output of OFFALU 20242 is connected to JPD Bus 10142, to a first input of AON Input Select Multiplexer (AONSEL) 20248 and AONP 20216, and, as described above, to a third input of OFFSEL 20238. A second 32 bit data output of OFFALU 20242 is connected to OFFSET Bus 20228 and third 16 bit output is connected to NAME Bus 20224.

20 **b. AON Processor 20216 Structure**

Referring to AONP 20216, a primary function of AONP 20216 is that of containing AON fields of AON pointers and logical descriptors. In addition, those portions of AONGRF 20232 not otherwise occupied by AON pointers and logical descriptors may be used as a 28 bit wide general register area by JP 10114. These 25 portions of AONGRF 20232 may be so used either alone or in conjunction with corresponding portions of OFFGRF 20234 and LENGRF 20236. AONP 20216 includes AONSEL 20248 and AONGRF 20232. As previously described, a first 32 bit data input AONSEL 20248 is connected from a first data output of OFFALU 20242. A second 28 bit data input of AONSEL 20248 is connected from 28 bit output of AONGRF 20232 and from AON Bus 20230. A third 28 bit data input of AONSEL 20248 is connected from logic zero, 30 that is a 28 bit input wherein each input bit is set to logic zero. Twenty-eight bit data output of AONSEL 20248 is connected to data input of AONGRF 20232. As just described, 28 bit data output of AONGRF 20232 is connected to second data input of AONSEL 20248, and is connected to AON Bus 20230.

35 **c. Length Processor 20220 Structure**

Referring finally to LENP 20220, a primary function of LENP 20220 is the generation manipulation of length fields of AON pointers and physical descriptors. In addition, LENGRF 20236 may be used, in part, either alone or in conjunction with corresponding address spaces of AONGRF 20232 and OFFGRF 20234, as general registers for storage of data. LENP 20220 includes Length Input Select Multiplexer (LENSEL) 20250, 40 LENGRF 20236, BIAS 20246, and Length ALU (LENALU) 20252. LENSEL 20250 has first and second data inputs connected from, respectively, LENGTH Bus 20226 and OFFSET Bus 20228. LENGTH Bus 20226 is eight data bits, zero filled while OFFSET Bus 20228 is 32 data bits. LENSEL 20250 has a third 32 bit data input connected from data output of LENALU 20252. Thirty-two bit data output of LENSEL 20250 is connected to data input of LENGRF 20236 and to a first data input of BIAS 20246. Second and third 32 bit data inputs of BIAS 20246 are connected from, respectively, Constant (C) and Literal (L) outputs of FUSITT 45 11012 as will be described further below. Thirty-two bits data output of LENGRF 20236 is connected to JPD Bus 10142, to Write Length Input (WL) input of NC 10226, and to a first input of LENALU 20252. Five bit output of BIAS 20246 is connected to a second input of LENALU 20252, to LENGTH Bus 20226, and, as previously described, to a fourth input of OFFMUX 20240. Thirty-two bit output of LENALU 20252 is connected, as stated above, to third input of LENSEL 20250.

50 Having described the overall operation and the structure of DESP 20210, operation of DESP 20210 will be described next below in further detail.

d. Descriptor Processor 20210 Operation

a.a. Offset Selector 20238

55 Referring to OFFP 20218, GRF 10354 includes GR's 10360 and SR's 10362. GR's 10360 in turn contain ABR's 10364, mCR's 10366, and a set of general registers. SR's 10362 include MIS 10368 and MOS 10370. GRF 10354 is vertically divided into three parts. AONGRF 20232 is 28 bits wide and resides in AONP 20216, LENGRF 10354 is 32 bits wide and resides in LENP 20220, and OFFGRF 20234 is 32 bits wide and resides in OFFP 20218. AONGRF 20232, OFFGRF 20234, and LENGRF 20236 may be comprised of Fairchild 93422s.

60 In addition to storing offset fields of AON pointers and logical descriptors, those portions of OFFGRF 20234 not reserved for ABR's 10365, mCR's 10366, and SR's 10362 may be used as general registers, alone or in conjunction with corresponding portions AONGRF 20232 and LENGRF 20236, when OFFP 20218 is being utilized as a general purpose, 32 bit CPU. OFFGRF 20234 as will be described further below, is addressed in parallel with AONGRF 20232 and LENGRF 20236 by address inputs provided from FUCTL 65 20214.

OFFSEL 20238 is a multiplexer, comprised for example of SN74S244s and SN74S257s, for selecting data inputs to be written into selected address locations of OFFGRF 20234. OFFSEL 20238's first data input is from MOD Bus 10144 and is the primary path for data transfer between MEM 10112 and DESP 20210. As previously described, each data read from MEM 10112 to JP 10114 is a single 32 bit word where between one and 32 bits may contain actual data. If a data item to be read from MEM 10112 contains more than 32 bits of data, successive read operations are performed until the entire data item has been transferred.

OFFSEL 20238's second data input is from JPD Bus 10142. As will be described further below, JPD Bus 10142 is a data transfer path by which data outputs of FU 10120 and EU 10122 are written into MEM 10112. OFFSEL 20238's input of JPD Bus 10142 thereby provides a wrap around path by which data present at outputs of FU 10120 or EU 10122 may be transferred back into DESP 20210 for further use. For example, as previously stated a first output of OFFALU 20242 is connected to JPD Bus 10142, thereby allowing data output of OFFP 20218 to be returned to OFFP 20218 for further processing, or to be transferred to AONP 20216 or LENP 20220 as will be described further below. In addition, output of LENGRF 20236 is also connected to JPD Bus 10142 so that length fields of AON pointers or physical descriptors, or data, may be read from LENGRF 20236 to OFFP 20218. This path may be used, for example, when LENGRF 20236 is being used as a general purpose register for storing data or intermediate results of arithmetic or logical operations.

OFFSEL 20238's third input is provided from OFFALU 20242's output. This data path thereby provides a wrap around path whereby offset fields or data residing in OFFGRF 20234 may be operated on and returned to OFFGRF 20234, either in the same address location as originally read from or to a different address location. OFFP 20218 wrap around path from OFFALU 20242's output to OFFSEL 20238's third input, and thus to OFFGRF 20234, may be utilized, for example, in reading from MEM 10112 a data item containing more than 32 bits of data. As previously described, each read operation from MEM 10112 to JP 10114 is of a 32 bit word wherein between one and 32 bits may contain actual data. Transfer of a data word containing more than 32 bits is accomplished by performing a succession of read operations from MEM 10112 to JP 10114. For example, if a requested data item contains 70 bits of data, that data item will be transferred in three consecutive read operations. First and second read operations will each transfer 32 bits of data, and final read operation will transfer the remaining 6 bits of data. To read a data item of greater than 32 bits from MEM 10112 therefore, DESP 20210 must generate a sequence of logical descriptors, each defining a successive 32 bit segment of that data item. Final logical descriptor of the sequence may define a segment of less than 32 bits, for example, six bits as in the example just stated. In each successive physical descriptor, offset field must be incremented by value of length field of the preceding physical descriptor to define starting addresses of successive data items segments to be transferred. Length field of succeeding physical descriptors will, in general, remain constant at 32 bits except for final transfer which may be less than 32 bits. Offset field will thereby usually be incremented by 32 bits at each transfer until final transfer. OFFP 20218's wrap around data path from OFFALU 20242's output to their input of OFFSEL 20238 may, as stated above, be utilized in such sequential data transfer operations to write incremented or decremented offset field of a current physical descriptor back into OFFGRF 20234 to be offset field of a next succeeding physical descriptor.

In a further example, OFFP 20218's wrap around path from OFFALU 20242's output to third input of OFFSEL 20238 may be used in resolving Entries in Name Tables 10350, that is Name resolutions. In Name resolutions, as previously described, offset fields of AON pointers, for example Linkage Pointers 10416, are successively added and subtracted to provide a final AON pointer to a desired data item.

OFFSEL 20238's fourth input, from AONGRF 20232's output, may be used to transfer data or AON fields from AONGRF 20232 to OFFGRF 20234 or OFFMUX 20240. This data path may be used, for example, when OFFP 20218 is used to generate AON fields of AON pointers or physical descriptors or when performing Name evaluations.

Finally, OFFSEL 20238's fifth data input from OFFSET Bus 20228 allows offset fields on OFFSET Bus 20228 to be written into OFFGRF 20234 or transferred into OFFMUX 20240. This data path may be used, for example, to copy offset fields to OFFGRF 20234 when JP 10114 is performing a Name evaluation.

Referring now to OFFMUX 20240, OFFMUX 20240 includes logic circuitry for manipulating individual bits of 32 bit words. OFFMUX 20240 may be used, for example, to increment and decrement offset fields by length fields when performing string transfers, and to generate entries for, for example, MHT 10716 and MFT 10718. OFFMUX 20240 may also be used to aid in generating and manipulating AON, OFFSET, and LENGTH fields of physical descriptors and AON pointers.

b.b. Offset Multiplexer 20240 Detailed Structure (Fig. 238)

Referring to Fig. 238, a more detailed, partial block diagram of OFFMUX 20240 is shown. OFFMUX 20240 includes Offset Multiplexer Input Selector (OFFMUXIS) 23810, which for example may be comprised of SN74S373s and SN74S244s and Offset Multiplexer Register (OFFMUXR) 23812, which for example may be comprised of SN74S374s. OFFMUX 20240 also includes Field Extraction Circuit (FEXT) 23814, which may for example be comprised of SN74S257s, and Offset Multiplexer Field Selector (OFFMUXFS) 23816, which for example may be comprised of SN74S257s and SN74S374s. Finally, OFFMUX 20240 includes Offset Scaler (OFFSCALE) 23818, which may for example be comprised of AMD 25S10s, Offset Inter-element Spacing Encoder (OFFIESENC) 23820, which may for example be comprised of Fairchild 93427s

and Offset Multiplexer Output Selector (OFFMUXOS) 23822, which may for example be comprised of AMD 25Ss, Fairchild 93427s, and SN74S244s.

Referring first to OFFMUX 20240's connections to other portions of OFFP 20218, OFFMUX 20240's first data input, from OFFSEL 20238, is connected to a first input of OFFMUXIS 23810. OFFMUX 20240's second input, from MOD Bus 10144, is connected to a second input of OFFMUXIS 23810. OFFMUX 20240's third input, from JPD Bus 10142, is connected to a first input of OFFMUXFS 23816 while OFFMUX 20240's fourth input, from BIAS 20246, is connected to a first input of OFFMUXOS 23822. OFFMUX 20240's fifth input, from NAME Bus 20224, is connected to a second input of OFFMUXFS 23816. OFFMUX 20240's first output, to OFFALUSA 20244, is connected from output of OFFMUXR 23812 while OFFMUX 20240's second output, to OFFALU 20242, is connected from output of OFFMUXOS 23822.

Referring to OFFMUX 20240's internal connections, 32 bit output of OFFMUXIS 23810 is connected to input OFFMUXR 23812 and 32 bit output of OFFMUXR 23812 is connected, as described above, as first output of OFFMUX 20240, and as a third input of OFFMUXFS 23816. Thirty-two bit output of OFFMUXR 23812 is also connected to input of FEXT 23814. OFFMUXFS 23816's first, second and third inputs are connected as described above. A fourth input of OFFMUXFS 23816 is a 32 bit input wherein 31 bits are set to logic zero and 1 bit to logic 1. A fifth input is a 32 bit input wherein 31 bits are set to logic 1 and 1 to logic 0. A sixth input of OFFMUXFS 23816 is a 32 bit literal (L) input provided from FUSITT 11012 and is a 32 bit binary number comprising a part of a microinstruction FUCTL 20214, described below. OFFMUXFS 23816's seventh and eighth input are connected from FEXT 23814. Input 7 comprises FIU and TYPE fields of Name Table Entries which have been read into OFFMUXR 23812. Input 8 is a general purpose input conveying bits extracted from a 32 bit word captured in OFFMUXR 23812. As indicated in Fig. 238, OFFMUXFS 23816's first, third, fourth, fifth, and sixth inputs are each 32 bit inputs which are divided to provide two 16 bit inputs each. That is, each of these 32 bit inputs is divided into a first input comprising bit 0 to 15 of that 32 bit input, and a second input comprising bits 16 to 31.

Thirty-two bit output of OFFMUXFS 23816 is connected to inputs of OFFSCALE 23818 and OFFIESENC 23820. As indicated in Fig. 238, Field Select Output (FSO) of OFFMUXFS 23816 is a 32 bit word divided into a first word including 0 to 15 and a second word including bits 16 to 31. Output FSO of OFFMUXFS 23816, as will be described further below, thereby reflects the divided structure of OFFMUXFS 23816's first, third, fourth, fifth, and sixth inputs.

Logical functions performed by OFFMUXFS 23816 in generating output FSO, and which will be described in further detail in following descriptions, include:

- (1) Passing the contents of OFFMUXR 23812 directly through OFFMUXFS 23816;
- (2) Passing a 32 bit word on JPD Bus 10142 directly through OFFMUXFS 23816;
- (3) Passing a literal value comprising a part of a microinstruction from FUCTL 20214 directly through OFFMUXFS 23816;
- (4) Forcing FSO to be literal values 0000 0000;
- (5) Forcing FSO to be literal value 0000 001;
- (6) Extracting Name Table Entry fields;
- (7) Accepting a 32 bit word from OFFMUXR 23812 or JPD Bus 10142, or 32 bits of a microinstruction from FUCTL 20214, and passing the lower 16 bits while forcing the upper 16 bits to logic 0;
- (8) Accepting a 32 bit word from OFFMUXR 23812 or JPD Bus 10142, or 32 bits of microinstruction from FUCTL 20214, and passing the higher 16 bits while forcing the lower 16 bits to logic 0;
- (9) Accepting a 32 bit word from OFFMUXR 23812, or JPD Bus 10142, or Name Bus 20224, or 32 bits of a microinstruction from FUCTL 20214, and passing the lower 16 bits while sign extending bit 16 to the upper 16 bits; and,
- (10) Accepting a 32 bit word from Name Bus 20224 and passing the lowest 8 bits while sign extending bit 24 to the highest 24 bits.

Thirty-two bit output of OFFSCALE 23818 and 3 bit output of OFFIESENC 23820 are connected, respectively, to second and third inputs of OFFMUXOS 23822. OFFMUXOS 23822's first input is, as described above, OFFMUX 20240's fourth input and is connected from output BIAS 20246. Finally, OFFMUXOS 23822's 32 bit output, OFFMUX (0—31) is OFFMUX 20240's second output and as previously described as connected to a second input of OFFALU 20242.

c.c. Offset Multiplexer 20240 Detailed Operation

a.a.a. Internal Operation

Having described the structure of OFFMUX 20240 as shown in Fig. 238, operation of OFFMUX 20240 will be described below. Internal operation of OFFMUX 20240, as shown in Fig. 238, will be described first, followed by description of OFFMUX 20240's operation with regard to DESP 20210.

Referring first to OFFMUXR 23812, OFFMUXR 23812 is a 32 bit register receiving either a 32 bit word from MOD Bus 10144, MOD (0—31), or a 32 bit word received from OFFSEL 20238, OFFSEL (0—31), and is selected by OFFMUXIS 23810. OFFMUXR 23812 in turn provides those selected 32 bit words from MOD Bus 10144 or OFFSEL 20238 as OFFMUX 20240's first data output to OFFALUSA 20244, as FEXT 23814's input, and as OFFMUXFS 23816's third input. OFFMUXR 23812's 32 bit output to OFFMUXFS 23816 is provided as two parallel 16 bit words designated as OFFMUXR output (OFFMUXRO) (0—15) and (16—31). As described above, OFFMUXFS 23816's output to OFFALUSA 20244 from OFFMUXR 23812 may be right shifted 16

places and the highest 16 bits zero filled.

FEXT 23814 receives OFFMUXRO (0—15) and (16—31) from OFFMUXR 23812 and extracts certain fields from those 16 bit words. In particular, FEXT 23814 extracts FIU and TYPE fields from NT 10350 Entries which have been transferred into OFFMUXR 23812. FEXT 23814 may then provide those FIU and TYPE fields as OFFMUXFS 23816's seventh input. FEXT 23814 may, selectively, extract certain other fields from 32 bit words residing in OFFMUXR 23812 and provide those fields as OFFMUXFS 23816's eighth input.

OFFMUXFS 23816 operates as a multiplexer to select certain fields from OFFMUXFS 23816's eight inputs and provide corresponding 32 bit output words, Field Select Output (FSO), comprised of those selected fields from OFFMUXFS 23816's inputs. As previously described, FSO is comprised of 2, parallel 16 bit words, FSO (0—15) and FSO (16—31). Correspondingly, OFFMUX 20240's third input, from JPD Bus 10142, is a 32 bit input presented as two 16 bit words, JPD (0—15) and JPD (16—31). Similarly, OFFMUXFS 23816's fourth, fifth, and sixth inputs are each presented as 32 bit words comprised of 2, parallel 16 bit words, respectively, "0" (0—15) and (16—31), "1" (0—15) and (16—31), and L (0—15) and (16—31). OFFMUXFS 23816's second input, from NAME Bus 20224, is presented as a single 16 bit word, NAME (16—31), while OFFMUXFS 23816's inputs from FEXT 23814 are each less than 16 bits in width. OFFMUXFS 23816 may, for a single 32 bit output word, select FSO (0—15) to contain one of corresponding 16 bit inputs JPD (0—15), "0" (0—15), "1" (0—15), or L (0—15). Similarly, FSO (16—31) of that 32 bit output word may be selected to contain one of NAME (16—31), JPD (16—31), O (16—31), 1 (16—31), L (16—31), or inputs 7 and 8 from FEXT 23814. OFFMUXFS 23816 therefore allows 32 bit words, comprised of two 16 bit fields, to be generated from selected portions of OFFMUXFS 23816's inputs.

OFFMUXFS 23816 32 bit output is provided as inputs to OFFSCALE 23818 and OFFIESENC 23820. Referring first to OFFIESENC 23820, OFFIESENC 23820 is used, in particular, in resolving, or evaluating, NT 10350 Entries (NTEs) referring to arrays of data words. As indicated in Fig. 108, word D of an NTE contains certain information relating to inter-element spacing (IES) of data words of an array. Word D of an NTE may be read from MEM 10112 to MOD Bus 10144 and through OFFMUX 20240 to input of OFFIESENC 23820. OFFIESENC 23820 then examines word D's IES field to determine whether inter-element spacing of that array is a binary multiple, that is 1, 2, 4, 8, 16, 32, or 64 bits. In particular, OFFIESENC 23820 determines whether 32 bit word D contains logic zeros in the most significant 25 bits and a single logic one in the least significant 7 bits. If inter-element spacing is such a binary multiple, starting addresses of data words of that array may be determined by left shifting of index (IES) to obtain offset fields of physical addresses of words in the array and a slower and more complex multiplication operation is not required. In such cases, OFFIESENC generates a first output, IES Encodeable (IESENC) to FUCTL 20214 to indicate that inter-element spacing may be determined by simple left shifting. OFFIESENC 23820 then generates encoded output, Encoded IES (ENCIES), to OFFMUXOS 23822. ENCIES is then a coded value specifying the amount of left shift necessary to translate index (IES) value into offsets of words in that array. As indicated in Fig. 238, ENCIES is OFFMUXOS 23822's third input.

OFFSCALE 23818 is a left shift network with zero fill of least significant bits, as bits are left shifted. Amount of shift by OFFSCALE 23818 is selectable between zero and 7 bits. Thirty-two bit words transferred into OFFSCALE 23818 from OFFSCALE 23818 from OFFMUXFS 23816 may therefore be left shifted, bit by bit, to selectively reposition bits within that 32 bit input word. In conjunction with OFFMUXFS 23816, and a wrap around connection provided by OFFALU 20242's output to OFFSEL 20238, OFFSCALE 23818 may be used to generate and manipulate, for example, entries for MHT 10716, MFT 10718, AOT 10712, and AST 10914, and other CS 10110 data structures.

OFFMUXOS 23822 is a multiplexer having first, second, and third inputs from, respectively, BIAS 20246, OFFSCALE 23818, OFFIESENC 23820. OFFMUXOS 23822 may select any one of these inputs as OFFMUX 20240's second output, OFFMUX (0—31). As previously described, OFFMUX 20240's second output is connected to a second input of OFFALU 20242.

Having described internal of OFFMUX 20240, operation of OFFMUX 20240 with regard to overall operation of DESP 20210 will be described next below.

b.b.b. Operation Relative to Descriptor Processor 20210

OFFMUX 20240's first input, from OFFSEL 20238, allows inputs to OFFSEL to be transferred through OFFMUXIS 23810 and into OFFMUXR 23812. This input allows OFFMUXR 23812 to be loaded, under control of FUCTL 20214 microinstructions, with any input of OFFSEL 20238. In a particular example, OFFALU 20242's output may be fed back through OFFSEL 20238's third input and OFFMUX 20240's first input to allow OFFMUX 20240 and OFFALU 20242 to perform reiterative operations on a single 32 bit word.

OFFMUX 20240's second input, from MOD Bus 10144, allows OFFMUXR 23812 to be loaded directly from MOD Bus 10144. For example, NTEs from a currently active procedure may be loaded into OFFMUXR 23812 to be operated upon as described above. In addition, OFFMUX 20240's second input may be used in conjunction with OFFSEL 20238's first input, from MOD Bus 10144, as parallel input paths to OFFP 20218. These parallel input paths allow pipelining of OFFP 20218 operations by allowing OFFSEL 20238 and OFFGRF 20234 to operate independently from OFFMUX 20240. For example, FU 10120 may initiate a read operation from MEM 10112 to OFFMUXR 23812 during a first microinstruction. The data so requested will appear on MOD Bus 10144 during a second microinstruction and may be loaded into OFFMUXR 23812 through OFFMUX 20240's second input from MOD Bus 10144. Concurrently, FU 10120 may initiate, at start

of second microinstruction, an independent operation to be performed by OFFSEL 20238 and OFFGRF 20234, for example loading output of OFFALU 20242 into OFFGRF 20234. Therefore, by providing an independent path into OFFMUX 20240 from MOD Bus 10144, OFFSEL 20238 is free to perform other, concurrent data transfer operations while a data transfer from MOD Bus 10144 to OFFMUX 20240 is being performed.

OFFMUX 20240's third input, from JPD Bus 10142, is a general purpose data transfer path. For example, data from LENGRF 20236 or OFFALU 20242 may be transferred into OFFMUX 20240 through JPD Bus 10142 and OFFMUX 20240's third input.

OFFMUX 20240's fourth input is connected from BIAS 20246 and primarily used during string transfers as described above. That is, length fields of physical descriptors generated for a string transfer may be transferred into OFFMUX 20240 through OFFMUX 20240's fourth input to increment or decrement, offset fields of those physical descriptors in OFFALU 20242.

OFFMUX 20240's fifth input is connected from NAME Bus 20224. As will be described further below, Names are provided to NC 10226 by FUCTL 20214 to call, from MC 10226, logical descriptors corresponding to Names appearing on MOD Bus 10144 as part of sequences of SINS.

As each Name is presented to NC 10226, that Name is transferred into and captured in Name Trap (NT) 20254. Upon occurrence of an NC 10226 miss, that is NC 10226 does not contain an entry corresponding to a particular Name, that Name is subsequently transferred from NT 20254 to OFFMUX 20240 through NAME Bus 20224 and OFFMUX 20240's fifth input. That Name, which is previously described as an 8, 12, or 16 bit binary number, may then be scaled, that is multiplied by a NTE size. That scaled Name may then be added to Name Table Pointer (NTP) from mCRs 10366 to obtain the address of a corresponding NTE in an NT 10350. In addition, a Name resulting in a NC 10226 miss, or a page fault in the corresponding NT 10350, or requiring a sequence of Name resolves, may be transferred into OFFGRF 20234 from OFFMUX 20240, through OFFALU 20242 and OFFSEL 20238 third input. That Name may subsequently be read, or restored, from OFFGRF 20234 as required.

Referring now to outputs of OFFMUX 20240, OFFMUX 20240's first output, from OFFMUXR 23812, allows contents of OFFMUXR 23812 to be transferred to first input of OFFALU 20242 through OFFALUSA 20244. OFFMUX 20240's second output, from OFFMUXOS 23822, is provided directly to second input of OFFALU 20242. OFFALU 20242 may be concurrently provided with a first input from OFFMUXR 23812 and a second input, for example a manipulated offset field, from OFFMUXOS 23822.

Referring to OFFALUSA 20244, OFFALUSA 20244 is a multiplexer. OFFALUSA 20244 may select either output of OFFGRF 20234 or first output of OFFMUX 20240 to be either first input of OFFALU 20242 or to be OFFP 20218's output to OFFSET Bus 20228. For example, an offset field from OFFGRF 20234 may be read to OFFSET Bus 20228 to comprise offset field of a current logical descriptor, and concurrently read into OFFALU 20242 to be incremented or decremented to generate offset field of a subsequent logical descriptor in a string transfer.

OFFALU 20242 is a general purpose, 32 bit arithmetic and logic unit capable of performing all usual ALU operations. For example, OFFALU 20242 may add, subtract, increment, or decrement offset fields of logical descriptors. In addition, OFFALU 20242 may serve as a transfer path for data, that is OFFALU 20242 may transfer input data to OFFALU 20242's outputs without operating upon that data. OFFALU 20242's first output, as described above, is connected to JPD Bus 10142, to third input of OFFSEL 20238, and to first input of AONSEL 20248. Data transferred or manipulated by OFFALU 20242 may therefore be transferred on to JPD Bus 10142, or wrapped around into OFFP 20218 through OFFSEL 20238 for subsequent or reiterative operations. OFFALU 20242's output to AONSEL 20248 may be used, for example, to load AON fields of AON pointers or physical descriptors generated by OFFP 20218 into AONGRF 20232. In addition, this data path allows FU 10120 to utilize AONGRF 20232 as, for example, a buffer or temporary memory space for intermediate or final results of FU 10120 operations.

OFFALU 20242's output to OFFSET Bus 20228 allows logical descriptor offset fields to be transferred onto OFFSET Bus 20228 directly from OFFALU 20242. For example, a logical descriptor offset field may be generated by OFFALU 20242 during a first clock cycle, and transferred immediately onto OFFSET Bus 20228 during a second clock cycle.

OFFALU 20242's third output is to NAME Bus 20224. As will be described further below, NAME Bus 20224 is address input (ADR) to NC 10226. OFFALU 20242's output to NAME Bus 20224 thereby allows OFFP 20218 to generate or provide addresses, that is Names, to NC 10226.

Having described operation of OFFP 20218, operation of LENP 20220 will be described next below.

e. Length Processor 20220 (Fig. 239)

Referring to Fig. 202, a primary function of LENP 20220 is generation and manipulation of logical descriptor length fields, including length fields of logical descriptors generated in string transfers. LENP 20220 includes LENGRF 20236, LENSEL 20250, BIAS 20246, and LENALU 20252. LENGRF 20236 may be comprised, for example, of Fairchild 93422s. LENSEL 20250 may be comprised of, for example, SN74S257s, SN74S157s, and SN74S244s, and LENALU 20252 may be comprised of, for example, SN74S381s.

As previously described, LENGRF 20236 is a 32 bit wide vertical section of GRF 10354. LENGRF 20236 operates in parallel with OFFGRF 20234 and AONGRF 20232 and contains, in part, length fields of logical descriptors. In addition, also as previously described, LENGRF 20236 may contain data.

LENSEL 20250 is a multiplexer having three inputs and providing outputs to LENGRF 20236 and first input of BIAS 20246. LENSEL 20250's first input is from Length Bus 20226 and may be used to write physical descriptor or length fields from LENGTH Bus 20226 into LENGRF 20236 or into BIAS 20246. Such length fields may be written from LENGTH Bus 20226 to LENGRF 20236, for example, during Name evaluation or resolve operations. LENSEL 20250's second input is from OFFSET Bus 20228. LENSEL 20250's second input may be used, for example, to load length fields generated by OFFP 20218 into LENGRF 20236. In addition, data operated upon by OFFP 20218 may be read into LENGRF 20236 for storage through LENSEL 20250's second input.

LENSEL 20250's third input is from output of LENALU 20252 and is a wrap around path to return output of LENALU 20252 to LENGRF 20236. LENSEL 20250's third input may, for example, be used during string transfers when length fields of a particular logical descriptor is incremented or decremented by LENALU 20252 and returned to LENGRF 20236. This data path may also, for example, be used in moving a 32 bit word from one location in LENGRF 20236 to another location in LENGRF 20236. As stated above, LENSEL 20250's output is also provided to first input BIAS and allows data appearing at first, second, or third inputs of LENSEL 20250 to be provided to first input of BIAS 20246.

BIAS 20246, as will be described in further detail below, generates logical descriptor length fields during string transfers. As described above, no more than 32 bits of data may be read from MEM 10112 during a single read operation. A data item of greater than 32 bits in length must therefore be transferred in a series, or string, of read operations, each read operation transferring 32 bits or less of data. String transfer logical descriptor length fields generated BIAS 20246 are provided to LENGTH Bus 20226, to LENALU 20252 second input, and to OFFMUX 20240's fourth input, as previously described. These string transfer logical descriptor length fields, referred to as bias fields are provided to LENGTH Bus 20226 by BIAS 20246 to be length fields of the series of logical descriptors generated by DESP 20210 to execute a string transfer. These bias fields are provided to fourth input OFFMUX 20240 to increment or decrement offset fields of those logical descriptors, as previously described. These bias fields are provided to second input of LENALU 20252, during string transfers, to correspondingly decrement the length field of a data item being read to MEM 10112 in a string transfer. BIAS 20246 will be described in greater detail below, after LENALU 20252 is first briefly described.

a.a. Length ALU 20252

LENALU 20252 is a general purpose, 32 bit arithmetic and logic unit capable of executing all customary arithmetic and logic operations. In particular, during a string transfer of a particular data item LENALU 20252 receives that data item's length field from LENGRF 20236 and successive bias fields from BIAS 20246. LENALU 20252 then decrements that logical descriptor's current length field to generate length field to be used during next read operation of the string transfer, and transfers new length field back into LENGRF 20236 through LENSEL 20250's third input.

b.b. BIAS 20246 (Fig. 239)

Referring to Fig. 239, a partial block diagram of BIAS 20246 is shown. BIAS 20246 includes Bias Memory (BIASM) 23910, Length Detector (LDET) 23912, Next Zero Detector (NXTZRO) 23914, and Select Bias (SBIAS) 23916. Input of LDET 23912 is first input of BIAS 20246 and connected from output of LENSEL 20250. Output of LDET 23912 is connected to data input of BIASM 23910, and data output of BIASM 23910 is connected to input of NXTZRO 23914. Output of NXTZRO 23914 is connected to a first input of SBIAS 23916. A second input of SBIAS 23916 is BIAS 20246's second input, LB, and is connected from an output of FUCTL 20214. A third input of SBIAS 23916 is BIAS 20246's third input, L, and is connected from yet another output of FUCTL 20214. Output of SBIAS 23916 is output of BIAS 20246 and, as described above, is connected to LENGTH Bus 20226, to a second input of LENALU 20252, and to fourth input of OFFMUX 20240.

BIASM 23910 is a 7 bit wide random access memory having a length equal to, and operating and addressed in parallel with, SR's 10362 of GRF 10354. BIASM 23910 has an address location corresponding to each address location of SR's 10362 and is addressed concurrently with those address locations in SR's 10362. BIASM 23910 may be comprised, for example, of AMD 27SO3As.

BIASM 23910 contains a bias value of each logical descriptor residing in SR's 10362. As described above, a bias value is a number representing number of bits to be read from MEM 10112 in a particular read operation when a data item having a corresponding logical descriptor, with a length field stored LENGRF 20236, is to be read from MEM 10112. Initially, bias values are written into BIASM 23910, in a manner described below, when their corresponding length fields are written into LENGRF 20236. If a particular data item has a length of less than 32 bits, that data item's initial bias value will represent that data item's actual length. For example, if a data item has a length of 24 bits the associated bias value will be a 6 bit binary number representing 24. That data item's length field in LENGRF 20236 will similarly contain a length value of 24. If a particular item has a length of greater than 32 bits for example, 70 bits as described in a previous example, that data item must be read from MEM 10112 in a string transfer operation. As previously described, a string transfer is a series of read operations transferring 32 bits at a time from MEM 10112, with a final transfer of 32 bits or less completing transfer of that data item. Such a data item's initial length field entry in LENGRF 20236 will contain, using the same example as previously described, a value of 70.

EP 0 067 556 B1

That data item's initial bias entry written into a corresponding address space of BIASM 23910 will contain a bias value of 32. That initial bias value of 32 indicates that at least the first read operation required to transfer that data item from MEM 10112 will transfer 32 bits of data.

When a data item having a length of less than 32 bits, for example 24 bits, is to be read from MEM 10112, that data item's bias value of 24 is read from BIASM 23910 and provided to LENGTH Bus 20226 as length field of logical descriptor for that read operation. Concurrently, that bias value of 24 is subtracted from that data item's length field read from LENGRF 20236. Subtracting that bias value from that length value will yield a result of zero, indicating that no further read operations are required to complete transfer of that data item.

If a data item having, for example, a length of 70 bits is to be read from MEM 10112, that data item's initial bias value of 32 is read from BIASM 23910 to LENGTH Bus 20226 as length field of first logical descriptor of a string transfer. Concurrently, that data item's initial length field is read from LENGRF 20236. That data item's initial bias value, 32, is subtracted from that data item's initial length value, 70, and LENALU 20252. The result of that subtraction operation is the remaining length of data item to be transferred in one or more subsequent read operations. In this example, subtracting initial bias value from initial length value indicates that 38 bits of that data item remain to be transferred. LENALU 20252's output representing results of this subtraction, for example 38, are transferred to LENSEL 20250's third input to LENGRF 20236 and written into address location from which that data item's initial length value was read. This new length field entry then represents remaining length of that data item. Concurrently, LDET 23912 examines that residual length value being written into LENGRF 20236 to determine whether remaining length of that data item is greater than 32 bits or is equal to or less than 32 bits. If remaining length is greater than 32 bits, LDET 23912 generates a next bias value of 32, which is written into BIASM 23910 and same address location that held initial bias value. If remaining data item length is less than 32 bits, LDET 23912 generates a 6 bit binary number representing actual remaining length of data item to be transferred. Actual remaining length would then, again, be written into BIASM 23910 address location originally containing initial bias value. These operations are also performed by LDET 23912 in examining initial length field and generating a corresponding initial bias value. These read operations are continued as described above until LDET 23912 detects that remaining length field is 32 bits or less, and thus that transfer of that data item will be completed upon next read operation. When this event is detected, LDET 23912 generates a seventh bit input into BIASM 23910, which is written into BIASM 23910 together with last bias value of that string transfer, indicating that remaining length will be zero after next read operation. When a final bias value is read from BIASM 23910 at start of next read operation of that string transfer, that seventh bit is examined by NXTZRO 23914 which subsequently generates a test condition output, Last Read (LSTRD) to FUCTL 20214. FUCTL 20214 may then terminate execution of that string transfer after that last read operation, if the transfer has been successfully completed.

As previously described, the basic unit of length of a data item in CS 10110 is 32 bits. Accordingly, data items of 32 or fewer bits may be transferred directly while data items of more than 32 bits require a string transfer. In addition, transfer of a data item through a string transfer requires tracking of the transferred length, and remaining length to be transferred, of both the data item itself and the data storage space of the location the data item is being transferred to. As such, BIAS 20246 will store, and operate with, in the manner described above, length and bias fields of the logical descriptors of both the data item and the location the data item is being transferred to. FUCTL 20214 will receive an LSTRD test condition if bias field of source descriptor becomes zero before or concurrently with that of the destination, that is a completed transfer, or if bias field of destination becomes zero before that of the source, and may provide an appropriate microcode control response. It should be noted that if source bias field becomes zero before that of the destination, the remainder of the location that this data item is being transferred to will be filled and padded with zeros. If the data item is larger than the destination storage capacity, the destination location will be filled to capacity and FUCTL 20214 notified to initiate appropriate action.

In addition to allowing data item transfers which are insensitive to data item length, BIAS 20246 allows string transfers to be accomplished by short, tight microcode loops which are insensitive to data item length. A string transfer, for example, from location A to location B is encoded as:

(1) Fetch from A, subtract length from bias A, and update offset and length of a; and,

(2) Store to B, subtract length from bias B, and branch to (1) if length of B does not go to zero or fall through (end transfer) if length of B goes to zero. Source (A) length need not be tested as the microcode loop continues until length of B goes to zero; as described above, B will be filled and padded with zeros if length of A is less than length of B, or B will be filled and the string transfer ended if length of A is greater than or equal to length of B.

LDET 23912 and NXTZRO 23914 thereby allow FUCTL 20214 to automatically initiate a string transfer upon occurrence of a single microinstruction from FUCTL 20214 initiating a read operation by DESP 20210. That microinstruction initiating a read operation will then be automatically repeated until LSTRD to FUCTL 20214 from NXTZRO 23914 indicates that the string transfer is completed. LDET 23912 and NXTZRO 23914 may, respectively, be comprised for example of S74S260s, SN74S133s, SN74S51s, SN74S00s, SN74S00s, SN74S04s, SN74S02s, and SN74S32s.

Referring finally to SBIAS 23916, SBIAS 23916 is a multiplexer comprised, for example, of SN74S288s, SN74S374s, and SN74S244s. SBIAS 23916, under microinstruction control from FUCTL 20214, selects BIAS

20246's output to be one of a bias value from BIASM 23910, L8, or L. SBIAS 23916's first input, from BIASM 23910, has been described above. SBIAS 23916's second input, L8, is provided from FUCTL 20214 and is 8 bits of a microinstruction provided from FUSITT 11012. SBIAS 23916's second input allows microcode selection of bias values to be used in manipulation of length and offset fields of logical descriptors by LENCALU 20252 and OFFALU 20242, and for generating entries to MC 10226. SBIAS 23916's third input, L, is similarly provided from FUCTL 20214 and is a decoded length value derived from portions of microinstructions in FUSITT 11012. These microcode length values represent certain commonly occurring data item lengths, for example length of 1, 2, 4, 8, 16, 32, and 64 bits. An L input representing a length of 8 bits, may be used for example in reading data from MEM 10112 on a byte by byte basis.

Having described operation of LENCALU 20220, operation of AONP 20216 will be described next below.

f. AON Processor 20216

a.a. AONGRF 20232

As described above, AONP 20216 includes AONSEL 20248 and AONGRF 20232. AONGRF 20232 is a 28 bit wide vertical section of GRF 10354 and stores AON fields of AON pointers and logical descriptors. AONSEL 20248 is a multiplexer for selecting inputs to be written into AONGRF 20232. AONSEL 20248 may be comprised, for example of SN74S257s. AONGRF 20232 may be comprised of, for example, Fairchild 93422s.

As previously described, AONGRF 20232's output is connected onto AON Bus 20230 to allow AON fields of AON pointers and logical descriptors to be transferred onto AON Bus 20230 from AONGRF 20232. AONGRF 20232's output, together with a bi-directional input from AON Bus 20230, is connected to a second input of AONSEL 20248 and to a fourth input of AONSEL 20238. This data path allows AON fields, either from AONGRF 20232 or from AON Bus 20230, to be written into AONGRF 20232 or AONGRF 20234, or provided as an input to OFFMUX 20240.

b.b. AON Selector 20248

AONSEL 20248's first input is, as previously described, connected from output of OFFALU 20242 and is used, for example, to allow AON fields generated or manipulated by OFFP 20218 to be written into AONGRF 20232. AONSEL 20248's third input is a 28 bit word wherein each bit is a logical zero. AONSEL 20248's third input allows AON fields of all zeros to be written into AONGRF 20232. An AON field of all zeros is reserved to indicate that corresponding entries in OFFGRF 20234 and LENGRF 20236 are neither AON pointers nor logical descriptors. AON fields of all zeros are thereby reserved to indicate that corresponding entries in OFFGRF 20234 and LENGRF 20236 contain data.

In summary, as described above, DESP 20210 includes AONP 20216, OFFP 20218, and LENCALU 20220. OFFP 20218 contains a vertical section of GRF 10354, OFFGRF 20234, for storing offset fields of AON pointers and logical descriptors, and for containing data to be operated upon by DESP 20210. OFFP 20218 is principal path for transfer of data from MEM 10112 to JP 10114 and is a general purpose 32 bit arithmetic and logic unit for performing all usual arithmetic and logic operations. In addition, OFFP 20218 includes circuitry, for example OFFMUX 20240, for generation and manipulation of AON, OFFSET, and LENGTH fields of logical descriptors and AON pointers. OFFP 20218 may also generate and manipulate entries for, for example, NC 10226, ATU 10228, PC 10234, AOT 10712, MHT 10716, MFT 10718, and other data and address structures residing in MEM 10112. LENCALU 20220 includes a vertical section of GRF 10354, LENGRF 20236, for storing length fields of logical descriptors, and for storing data. LENCALU 20220 further includes BIAS 20246, used in conjunction with LENGRF 20236 and LENCALU 20252, for providing length fields of logical descriptors for MEM 10112 read operations and in particular automatically performing string transfers. AONP 20216 similarly includes a vertical section of GRF 10354, AONGRF 20232. A primary function AONGRF 20232 is storing and providing AON fields of AON pointers and logical descriptors.

Having described structure and operation of DESP 20210, structure and operation of Memory Interface (MEMINT) 20212 will be described next below.

2. Memory Interface 20212 (Figs. 106, 240)

MEMINT 20212 comprises FU 10120's interface to MEM 10112. As described above, MEMINT 20212 includes Name Cache (NC) 10226, Address Translation Unit (ATU) 10228, and Protection Cache (PC) 10234, all of which have been previously briefly described. MEMINT 20212 further includes Descriptor Trap (DEST) 20256 and Data Trap (DAT) 20258. Functions performed by MEMINT 20212 includes (1) resolution of Names to logical descriptors, by NC 10226; (2) translation of logical descriptors to physical descriptors, by ATU 10228; and (3) confirmation of access writes to objects, by PC 10234.

As shown in Fig. 202, NC 10226 address input (ADR) is connected from NAME Bus 20224. NC 10226 Write Length Field Input (WL) is connected from LENGRF 20236's output. NC 10226's Write Offset Field Input (WO) and Write AON Field Input (WA) are connected, respectively, from OFFSET Bus 20228 and AON Bus 20230. NC 10226 Read AON Field (RA), Read Offset Field (RO), and Read Length Field (RL) outputs are connected, respectively, to AON Bus 20230, OFFSET Bus 20228, and LENGTH Bus 20226.

DEST 20256's bi-directional AON (AON), Offset (OFF), and Length (LEN) ports are connected by bi-directional buses to and from, respectively, AON Bus 20230, OFFSET Bus 20228, and LENGTH Bus 20226. PC 10234 has AON (AON) and Offset (OFF) inputs connected from, respectively, AON Bus 20230 and

EP 0 067 556 B1

OFFSET Bus 20228. PC 10234 has a Write Entry (WEN) input connected from JPD Bus 10142. ATU 10228 has AON (AON), Offset (OFF), and Length (LEN) inputs connected from, respectively, AON Bus 20230, OFFSET Bus 20228, and LENGTH Bus 20226. ATU 10228's output is connected to physical Descriptor (PD) Bus 10146.

5 Finally, DAT 20258 has a bi-directional port connected to and from JPD Bus 10142.

a.a. Description Trap 20256 and Data Trap 20258

Referring first to DST 20256 and DAT 20258, DST 20256 is a register for receiving and capturing logical descriptors appearing on AON Bus 20230, OFFSET Bus 20228, and Length Bus 20226. Similarly, DAT 20258 is a register for receiving and capturing data words appearing on JPD Bus 10142. DST 20256 and DAT 20258 may subsequently return captured logical descriptors or data words to, respectively, AON Bus 20230, OFFSET Bus 20228, and LENGTH Bus 20226, and to JPD Bus 10142.

As previously described, many CS 10110 operations, in particular MEM 10112 and JP 10114 operations, are pipelined. That is, operations are overlapped with certain sets within two or more operations being executed concurrently. For example, FU 10120 may submit read request to MEM 10112 and, while MEM 10112 is accepting and servicing that request, submit a second read request. DEST 20256 and DAT 20258 assist in execution of overlapping operations by providing a temporary record of these operations. For example, a part of a read or write request to MEM 10112 by FU 10120 is a logical descriptor provided to ATU 10228. If, for example the first read request just referred to results in a ATU 10228 cache miss or a protection violation, the logical descriptor of that first request must be recovered for subsequent action by CS 10110 as previously described. That logical descriptor will have been captured and stored in DEST 20256 and thus is immediately available, so that DESP 20210 is not required to regenerate that descriptor. DAT 20258 serves a similar purpose with regard to data being written into MEM 10112 from JP 10114. That is, DAT 20258 receives and captures a copy of each 32 bit word transferred onto JPD Bus 10142 by JP 10114. In event of MEM 10112 being unable to accept a write request, that data may be subsequently reprovided from DAT 20258.

b.b. Name Cache 10226, Address Translation Unit 10228, and Protection Cache 10234 (Fig. 106)

Referring to NC 10226, ATU 10228, and PC 10234, these elements of MEMINT 20212 are primarily cache mechanisms to enhance the speed of FU 10120's interface to MEM 10112, and consequently of CS 10110's operation. As described previously, NC 10226 contains a set of logical descriptors corresponding to certain operand names currently appearing in a process being executed by CS 10110. NC 10226 thus effectively provides high speed resolution of certain operand names to corresponding logical descriptors. As described above with reference to string transfers, NC 10226 will generally contain logical descriptors only for data items of less than 256 bits length. NC 10226 read and write addresses are names provided on NAME Bus 20224. Name read and write addresses may be provided from DESP 20210, and in particular from OFFP 20218 as previously described, or from FUCTL 20214 as will be described in a following description of FUCTL 20214. Logical descriptors comprising NC 10226 entries, each entry comprising an AON field, an Offset field, a Length field, are written into NC 10226 through NC 10226 inputs WA, WO, and WL from, respectively, AON Bus 20230, OFFSET Bus 20228, and LENGRF 20236's output. Logical descriptors read from NC 10226 in response to names provided to NC 10226 ADR input are provided to AON Bus 20230, OFFSET Bus 20228, and LENGTH Bus 20226 from, respectively, NC 10226 outputs RA, RO, and RL.

ATU 10228 is similarly a cache mechanism for providing high speed translation of logical to physical descriptors. In general, ATU 10228 will contain, at any given time, a set of logical to physical page number mappings for MEM 10112 read and write requests which are currently being made, or anticipated to be made, to MEM 10112 by JP 10114. As previously described, each physical descriptor is comprised of a Frame Number (FN) field, and Offset Within Frame (O) fields, and a Length field. As discussed with reference to string transfers, a physical descriptor length field, as in a logical descriptor length field, specify a data item of less than or equal to 32 bits length. Referring to Fig. 106C, as previously discussed a logical descriptor comprised of a 14 bit AON field, a 32 bit Offset field, and Length field, wherein 32 bit logical descriptor Offset field is divided into a 18 bit Page Number (P) field and a 14 bit Offset within Page (O) field. In translating a logical into a physical descriptor, logical descriptor Length and O fields are used directly, as respectively, physical descriptor length and O fields. Logical descriptor AON and P fields are translated into physical descriptor FN field. Because no actual translation is required, ATU 10228 may provide logical descriptor L field and corresponding O field directly, that is without delay, to MEM 10112 as corresponding physical descriptor O and Length fields. ATU 10228 cache entries are thereby comprised of physical descriptor FN fields corresponding to AON and P fields of those logical descriptors for which ATU 10228 has corresponding entries. Because physical descriptor FN fields are provided from ATU 10228's cache, rather than directly as in physical descriptor O and Length fields, a physical descriptor's FN field will be provided to MEM 10112, for example, one clock cycle later than that physical descriptors O and Length fields, as has been previously discussed.

Referring to Fig. 202, physical descriptor FN fields to be written into ATU 10228 are, in general, generated by DESP 20210. FN fields to be written into ATU 10228 are provided to ATU 10228 Data Input (DI) through JPD Bus 10142. ATU 10228 read and write addresses are comprised of AON and P fields of logical.

descriptors and are provided to ATU 10228's AON and OFF inputs from, respectively, AON Bus 20230 and OFFSET Bus 20228. ATU 10228 read and write addresses may be provided from DESP 20210 or, as described further below, from FUCTL 20214. ATU 10228 FN outputs, together with O and Length fields comprising a physical descriptor, are provided to PD Bus 10146.

5 PC 10234 is a cache mechanism for confirming active procedure's access rights to objects identified by logical descriptors generated as a part of JP 10114 read or write requests to MEM 10112. As previously described access rights to objects are arbitrated on the basis of subjects. A subject has been defined as a particular combination of a principal, process, and domain. A principal, process, and domain are each identified by corresponding UIDs. Each subject having access rights to an object is assigned an Active
10 Subject Number (ASN) as described in a previous description of CS 10110's Protection Mechanism. The ASN of a subject currently active in CS 10110 is stored in ASN Register 10916 in FU 10120. Access rights of a currently active subject to currently active objects are read from those objects Access Control Lists (ACL) 10918 and stored in PC 10234. If the current ASN changes, PC 10234 is flushed of corresponding access right entries and new entries, corresponding to the new ASN, are written into PC 10234. The access rights
15 of a particular current ASN to a particular object may be determined by indexing, or addressing, PC 10234 with the AON identifying that object. Addresses to write entries into or read entries from PC 10234 are provided to PC 10234 AON input from AON Bus 20230. Entries to be written into PC 10234 are provided to PC 10234's WEN input from JPD Bus 10142. PC 10234 is also provided with inputs, not shown in Fig. 202 for purposes of clarity, from FUCTL 20214 indicating the current operation to be performed by JP 10114 with
20 respect to an object being presently addressed by FU 10120. Whenever FU 10120 submits a read or write request concerning a particular object to MEM 10112, AON field of that request is provided as an address to PC 10234. Access rights of the current active subject to that object are read from corresponding PC 10234 entry and compared to FUCTL 20214 inputs indicating the particular operation to be performed by JP 10114 with respect to that object. The operation to be performed by JP 10114 is then compared to that active
25 subject's access rights to that object and PC 10234 provides an output indicating whether that active subject possesses the rights required to perform the intended operation. Indexing of PC 10234 and comparison of access rights to intended operation is performed concurrently with translation of the memory request logical descriptor to a corresponding physical descriptor by ATU 10228. If PC 10234 indicates that that active subject has the required access rights, the intended operation is executed by JP 10114. If PC 10234
30 indicates that that active subject does not have the required access rights, PC 10234 indicates that a protection mechanism violation has occurred and interrupts execution of the intended operation.

c.c Structure and Operation of a Generalized Cache and NC 10226 (Fig. 240)

35 Having described overall structure and operation of NC 10226, ATU 10228, and PC 10234, structure and operation of these caches will be described in further detail below. Structure and operation of NC 10226, ATU 10228, and PC 10234 are similar, except that NC 10226 is a four-way set associative cache, ATU 10228 is a three-way set associative cache and PC 10234 is a two-way set associative cache.

40 As such, the structure and operation of NC 10226, ATU 10228, and PC 10234 will be described by reference to and description of a generalized cache similar but not necessarily identical to each of NC 10226, ATU 10228, and PC 10234. Reference will be made to NC 10226 in the description of a generalized cache next below, both to further illustrate structure and operation of the generalized cache, and to describe differences between the generalized cache and NC 10226. ATU 10228 and PC 10234 will then be described by description of differences between ATU 10228 and PC 10234 and the generalized cache.

45 Referring to Fig. 240, a partial block diagram of a generalized four-way, set associative cache is shown. Tag Store (TS) 24010 is comprised of Tag Store A (TSA) 24012, Tag Store B (TSB) 24014, Tag Store C (TSC) 24016, and Tag Store D (TSD) 24018. Each of the cache's sets, represented by TSA 24012 to TSD 24018, may contain, for example as in NC 10226, up to 16 entries, so that TSA 24012 to TSD 24018 are each 16 words long.

50 Address inputs to a cache are divided into a tag field and an index field. Tag fields are stored in the cache's tag store and indexed, that is addressed to be read or written from or to tag store by index field of the address. A tag read from tag store in response to index field of an address is then compared to tag field of that address to indicate whether the cache contains an entry corresponding to that address, that is, whether a cache hit occurs. In, for example, NC 10226, a Name syllable may be comprised of an 8, 12, or 16 bit binary word, as previously described. The four least significant bits of these words, or Names, comprise
55 NC 10226's index field while the remaining 4, 8, or 12 most significant bits comprise NC 10226's tag field. TSA 24012 to TSD 24018 may each, therefore, be 12 entry wide memories to store the 12 bit tag fields of 16 bit names. Index (IND) or address inputs of TSA 24012 to TSD 24018, would in NC 10226, be connected from four least significant bits of NAME Bus 20224 while Tag Inputs (TAGI) of TSA 24012 to TSD 24018 would be
60 connected from the 12 most significant bits of NAME Bus 20224.

65 As described above, tag outputs of TS 24010 are compared to tag fields of addresses presented to the cache to determine whether the cache contains an entry corresponding to that address. Using NC 10226 as an example 12 bit Tag Outputs (TAGOs) of TSA 24012 to TSD 24018 are connected to first inputs of Tag Store Comparators (TSC) 24019, respectively to inputs of Tag Store Comparitor A (TSCA) 24020, Tag Store Comparitor B (TSCB) 24022, Tag Store Comparitor D (TSCD) 24024, and Tag Store Comparitor E (TSCE) 24026. Second 12 bit inputs of TSCA 24020 to TSCE 24026 may be connected from the 12 most significant

EP 0 067 556 B1

bits of NAME Bus 20224 to receive tag fields of NC 10226 addresses. TAS 24020 to TSCE 24026 compare tag field of an address to tag outputs read from TSA 24012 to TSE 24018 in response to index field of that address, and provide four bit outputs indicating which, if any, of the possible 16 entries and their associated tag store correspond to that address tag field. TSCA 24020 to TSCE 24026 may be comprised, for example, of Fairchild 93S46s.

Four bit outputs of TSCA 24012 to TSCE 24026 are connected in the generalized cache to inputs of Tag Store Pipeline Registers (TSPR) 24027; respectively to inputs of Tag Store Pipeline Register A (TSPRA) 24028, Tag Store Pipeline Register B (TSPRB) 24030, Tag Store Pipeline Register C (TSPRC) 24032, and Tag Store Pipeline Register D (TSPRD) 24034. ATU 10228 and PC 10234 is pipelined with a single cache access operation being executed in two clock cycles. During first clock cycle tag store is addressed and tags store therein compared to tag field of address to provide indication of whether a cache hit has occurred, that is whether cache contains an entry corresponding to a particular address. During second clock cycle, as will be described below, a detected cache hit is encoded to obtain access to a corresponding entry in cache data store. Pipeline operation over two clock cycles is provided by cache pipeline registers which include, in part, TSPRA 24028 to TSPRD 24034. NC 10226 is not pipelined and does not include TSPRA 24028 to TSPRD 24034. In NC 10226, outputs of TSCA 24012 to TSCD 24024 are connected directly to inputs of TSHEA 24036 to TSHED 24042, described below.

Outputs of TSPRA 24028 to TSPRD 24034 are connected to inputs of Tag Store Hit Encoders (TSHE) 24035, respectively to Tag Store Hit Encoder A (TSHEA) 24036, Tag Store Hit Encoder B (TSHEB) 24038, Tag Store Hit Encoder C (TSHEC) 24040, and Tag Store Hit Encoder D (TSHED) 24042. TSHEA 24036 to TSHED 24042 encode, respectively, bit inputs from TSPRA 24028 to TSPRD 24034 to provide single bit outputs indicating which, if any, set of the cache's four sets includes an entry corresponding to the address input.

Single bit outputs of TSHEA 24036 to TSHED 24042 are connected to inputs of Hit Encoder (HE) 24044. HE 24044 encodes single bit inputs from TSHEA 24036 to TSHED 24042 to provide two sets of outputs. First outputs of HE 24044 are provided to Cache Usage Store (CUS) 24046 and indicate in which of the cache's four sets, corresponding to TSA 24012 to TSD 24018, a cache hit has occurred. As described previously with reference to MC 20116, and will be described further below, CUS 24046 is a memory containing information for tracking usage of cache entries. That is, CUS 24046 contains entries indicating whether, for a particular index, Set A, Set B, Set C or Set D of the cache's four sets has been most recently used and which has been least recently used. CUS 24046 entries regarding Sets A, B, C, and D are stored in, respectively, memories CUSA 24088, CUSB 24090, CUSC 24092, and CUSD 24094. Second output of HE 24044, as described further below, is connected to selection input of Data Store Selection Multiplexer (DSSMUX) 24048 to select an output from Data Store (DS) 24050 to be provided as output of the cache when a cache hit occurs.

Referring to DS 24050, as previously described a cache's data store contains the information, or entries, stored in that cache. For example, each entry in NC 10226's DS 24050 is a logical descriptor comprised of an AON, and Offset, and Length. A cache's data store parallels, in structure and organization, that cache's tag store and entries therein are identified and located through that cache's tag store and associated tag store comparison and decoding logic. In NC 10226, for example, for each Name having an entry in NC 10226 there will be an entry, the tag field of that name, stored in TS 24010 and a corresponding entry, a logical descriptor corresponding to that Name, in DS 24050. As described above, NC 10226 is a four-way, set associative cache so that TS 24010 and DS 24050 will each contain four sets of data. Each set was previously described as containing up to 16 entries. DS 24050 is therefore comprised of four 16 word memories. Each memory is 65 bits wide, accommodating 28 bits of AON, 32 bits of offset, and 5 bits of length. These four component data store memories of DS 24050 are indicated in Fig. 240 as Data Store A (DSA) 24052, Data Store B (DSB) 24054, Data Store C (DSC) 24056, and Data Store D (DSD) 24058. DSA 24052, DSB 24054, DSC 24056 and DSD 24058 correspond, respectively, in structure, contents, and operation to TSA 24012, TSB 24014, TSC 24016 and TSD 24018.

Data Inputs (Dis) of DSA 24052 to DSD 24058 are, in NC 10226 for example, connected from AON Bus 20230, OFFSET Bus 20228, LENGTH Bus 20226 and comprise inputs WA, WO, WL respectively of NC 10226. DSA 24052 to DSD 24058 Dis are, in NC 10226 as previously described, utilized in writing NC 10226 entries into DSA 24052 to DSD 24058. Address inputs of DSA 24052 to DSD 24058 are connected from address outputs of Address Pipeline Register (ADRPR) 24060. As will be described momentarily, except during cache flush operations, DSA 24052 to DSD 24058 address inputs are comprised of the same index fields of cache addresses as are provided as address inputs to TS 24010, but are delayed by one clock cycle and ADRPR 24060 for pipelining purposes. As described above, NC 10226 is not pipelined and does not have the one clock cycle delay. An address input to the cache will thereby result in corresponding entries, selected by index field of that address, being read from TSA 24012 to TSD 24018 and DSA 24052 to DSD 24058. The four outputs of DSA 24052 to DSD 24058 selected by a particular index field of a particular address are provided as inputs to DSSMUX 24048. DSSMUX 24048 is concurrently provided with selection control input from HE 24044. As previously described, this selection input to DSSMUX 24048 is derived from TS 24010 tag entries and indicates which of DSA 24052 to DSD 24058 entries corresponds to an address provided to the cache. In response to that selection control input, DSSMUX 24048 selects one of DS 24050's four logical descriptor outputs as the cache's output corresponding to that address. DSSMUX 24048's output is then provided, through Buffer Driver (BD) 24062 as the cache's output, for example in NC 10226 to AON Bus 20230, OFFSET Bus 20228, and LENGTH Bus 20226.

Referring to ADRMUX 24062, ADRMUX 24062 selects one of two sources to provide address inputs to DS 24050, that is to index to DS 24050. As described above, a first ADRMUX 24062 input is comprised of the cache's address index fields and, for example in NC 10226, is connected from the four least significant bits of NAME Bus 20224. During cache flush operations, DS 24050 address inputs are provided from Flush Counter (FLUSHCTR) 24066, which in the example is a four bit counter. During cache flush operations, FLUSHCTR 24066 generates sequential bit addresses which are used to sequentially address DSA 24052 to DSD 24058. Selection between ADRMUX 24062 first and second inputs, respectively the address index fields and from FLUSHCTR 24066, is controlled by Address Multiplexer Select (ADRMUXS) from FUCTL 20214.

Validity Store (VALS) 24068 and Dirty Store (DIRTYS) 24070 are memories operating in parallel with, and addressed in parallel with TS 24010. VALS 24068 contains entries indicating validity of corresponding TS 24010 and DS 24050 entries. That is, VALS 24068 entries indicate whether corresponding entries have been written into corresponding locations in TS 24010 and DS 24050. In the example, VALS 24068 may thereby be a 16 word by 4 bit wide memory. Each bit of a VALS 24068 word indicates validity of a corresponding location in TSA 24012 and DSA 24052, TSB 24014 and DSB 24054, TSC 24016 and DSC 24056, and TSD 24018 and DSD 24058. DIRTYS 24070 similarly indicates whether corresponding entries in corresponding locations of TS 24010 and DS 24050 have been written over, or modified. Again, DIRTYS 24070 will be a sixteen word by four bit wide memory.

Address inputs of VALS 24068 and DIRTYS 24070 are, for example in NC 10226, connected from least significant bits of NAME Bus 20224 and are thus addressed by index fields of NC 10226 addresses in parallel with TS 24010. Outputs of VALS 24068 are provided to TSCA 24020 to TSEE 24026 to inhibit outputs of TSCA 24020 through TSCE 24026 upon occurrence of an invalid concurrence between a TS 24010 entry and a NC 10226 address input. Similar outputs of DIRTYS 24070 are provided to FUCTL 20214 for use in cache flush operations to indicate which NC 10226 entries are dirty and must be written back into an MT 10350 rather than discarded.

Outputs of VALS 24068 and DIRTYS 24070 are also connected, respectively, to inputs of Validity Pipeline Register (VALPR) 24072 and Dirty Pipeline Register (DIRTYPR) 24074. VALPR 24072 and DIRTYPR 24074 are pipeline registers similar to TSPRA 24028 to TSPRD 24034 and are provided for timing purposes as will be described momentarily. Outputs of VALPR 24072 and DIRTYPR 24074 are connected to inputs of, respectively, Validity Write Logic (VWL) 24076 and Dirty Write Logic (DWL) 24078. As described above, NC 10226 is not a pipelined cache and does not include VALPR 24072 and DIRTYPR 24074; outputs of VALS 24068 and DIRTYS 24070 are connected directly to inputs of VWL 24076 and DWL 24078. Outputs of VWL 24076 and DWL 24078 are connected, respectively, to data inputs of VALS 24068 and DIRTYS 24070. Upon occurrence of a write operation to TS 24010 and DS 24050, that is writing in or modifying a cache entry, corresponding validity and dirty word entries are read from VALS 24068 and DIRTYS 24070 by index field of the caches input address. Outputs to VALS 24068 DIRTYS 24070 are received and stored in, respectively, VALPR 24072 and DIRTYPR 24074. At start of next clock cycle, validity and dirty words in VALPR 24072 and DIRTYPR 24074 are read into, respectively, VWL 24076 and DWL 24078. VWL 24076 and DWL 24078 respectively modify those validity or dirty word entries from VALS 24068 and DIRTYS 24070 in accordance to whether the corresponding entries in TS 24010 and DS 24050 are written into or modified. These modified validity and dirty words are then written, during second clock cycle, from VWL 24076 and DWL 24078 into, respectively, VALS 24068 and DIRTYS 24070. Control inputs of VWL 24076 and DWL 24078 are provided from FUCTL 20214.

Referring finally to Least Recent Used Logic (LRUL) 24080, LRUL 24080 tracks usage of cache entries. As previously described, the generalized cache of Fig. 240 is a four way, set associative cache with, for example, up to 16 entries in each of NC 10226's sets. Entries within a particular set are identified, as described above, by indexing the cache's TS 24010 and DS 24050 may contain, concurrently, up to four individual entries identified by the same index but distinguished by having different tags. In this case, one entry would reside in Set A, comprising TSA 24012 and DSA 24052, one in Set B, comprising TSB 24014 and DSB 24054, and so on. Since the possible number of individual entries having a common tag is greater than the number of cache sets, it may be necessary to delete a particular cache entry when another entry having the same tag is to be written into the cache. In general, the cache's least recently used entry would be deleted to provide a location in TS 24010 and DS 24050 for writing in the new entry. LRUL 24080 assists in determining which cache entries are to be deleted when necessary in writing in a new entry by tracking and indicating relative usage of the cache's entries. LRUL 24080 is primarily comprised of a memory, LRU Memory (MLRU) 24081, containing a word for each cache set. As described above, NC 10226, for example, includes 16 sets of 4 frames each, so that LRUL 24080's memory may correspondingly be, for example, 16 words long. Each word indicates relative usage of the 4 frames in a set and is a 6 bit word.

Words are generated and written into LRUL 24080's MLRU 24081, through input Register A, B, C, D (RABCD) 24083, according to a write only algorithm executed by HE 24044, as described momentarily. Each bit of each six word pertains to a pair of frames within a particular cache set and indicates which of those two frames was more recently used than the other. For example, Bit 0 will contain logic 1 if Frame A was used more recently than Frame B and a logic zero if Frame B was used more recently than Frame A. Similarly, Bit 1 pertains to Frames A and C, Bit 2 to Frames A and D, Bit 3 to Frames B and C, Bit 4 to Frames B and D, and Bit 5 to Frames C and D. Initially, all bits of a particular LRUL 24080 word are set to zero.

Assuming, for example, that the frames of a particular set are used in the sequence Frame A, Frame D, Frame B; Bits 0 to 5 of that LRUL 24080 word will initially contain all zeros. Upon a reference to Frame A, Bits 0, 1, and 2, referring respectively to Frames A and B, Frames A and C, and Frames A and D, will be written as logic 1's. Bits 3, 4, and 5, referring respectively to Frames B and C, Frames B and D, and Frames C and D, will remain logic 0. Upon reference to Frame D, Bits 0 and 1, referring respectively to Frames A and B and Frames A and C, will remain logic 1's. Bit 2, referring to Frames A and D, will be changed from logic 1 to logic 0 to indicate that Frame D has been referred to more recently than Frame A. Bit 3, referring to Frames B and C, will remain logic 0. Bits 4 and 5, referring respectively to Frames B and D and Frames C and D, will be written as logic 0, although they are already logic zeros, to indicate respectively that Frame D has been used more recently than Frame B or Frame C. Upon reference to Frame B, Bit 0, referring to Frames A and B, will be written to logic 0 to indicate that Frame B has been used more recently than Frame A. Bits 1 and 2, referring respectively to Frames A and C and Frames A and D, will remain respectively as logic 1 and logic 0. Bits three and four, referring respectively to Frames B and C and Frames B and D, will be written as logics 1's to indicate respectively that Frame B has been used more recently than Frame C or Frame D. Bit five will remain logic 0.

When it is necessary to replace a cache entry in a particular frame, the LRUL 24080 word referring to the cache set containing that frame will be read from LRUL 24080's MLRL 24081 through LRU Register (RLRU) 24085 and decoded by LRU Decode Logic (LRUD) 24087 to indicate which is least recently used frame. This decoding is executed by means of a Read Only Memory operating as a set of decoding gating.

Having described the structure and operation of a generalized cache as shown in Fig. 240, with references to NC 10226 for illustration and to point out differences between the generalized cache and NC 10226, structure and operation of ATU 10228 and PC 10234 will be described next below. ATU 10228 and PC 10234 will be described by describing the differences between ATU 10228 and PC 10234 and the generalized cache and NC 10226. ATU 10228 will be described first, followed by PC 10234.

d.d. Address Translation Unit 10228 and Protection Cache 10234

ATU 10228 is a three-way, set associative cache of 16 sets, that is contains 3 frames for each set. Structure and operation of ATU 10228 is similar to the generalized cache described above. Having 3 rather than 4 frames per set, ATU 10228 does not include a STD 24018, ATSCE 24026, ATSPRD 24034, ATSHED 24042, or ADSD 24058. As previously described ATU 10228 address inputs comprise AON and O fields of logical descriptors. AON fields are each 28 bits and O fields comprise the 18 most significant bits of logical descriptor offset fields, so that ATU 10228 address inputs are 48 bits wide. Four least significant bits of O fields are used as index. AON fields and the 14 most significant bits of O field comprise ATU 10228's tags. ATU 10228 tags are thereby each 42 bits in width. Accordingly, TSA 24012, TSB 24014, and TSC 24016 of ATU 10228's TS 24010 are each 16 words long by 42 bits wide.

DSA 24052, DSB 24054, and DSC 24056 of ATU 10228 are each 16 bits long. ATU 10228 outputs are, as previously described, physical descriptor Frame Number (FN) fields of 13 bits each. ATU 10228's DSA 24052, DSB 24054, DSC 24056 are thereby each 13 bits wide.

ATU 10228's LRUL 24080 is similar in structure and operation to that of the generalized cache. ATU 10228's LRUL 24080 words, each corresponding to an ATU 10228 set, are each 3 bits in width as 3 bits are sufficient to indicate relative usage of frames within a 3 frame set. In ATU 10228, Bit 1 of an LRUL 24080 word indicates whether Frame A was used more recently than Frame B, Bit 2 whether Frame A was used more recently than Frame C, and Bit 3 whether Frame B was used more recently than Frame C. In all other respects, other than as stated above, ATU 10228 is similar in structure and operation to the generalized cache.

Referring to PC 10234, PC 10234 is a two-way, set associative cache of 8 sets, that is has two frames per set. Having 2 rather than 4 frames, PC 10234 will not include a TSC 24016, a TSD 24018, a TSCC 24024, a TSCD 24026, a TSPRC 24032, a TSPRD 24034, a TSHED 24042, a TSHED 24042, a DSC 24056, or a DSD 24058.

Address inputs of PC 10234 are the 28 bit AON fields of logical descriptors. The 3 least significant bits of those AON fields are utilized as indexes for addressing PC 10234's TS 24010 and DS 24050. The 25 most significant bits of those AON field address inputs are utilized as PC 10234's tags, so that PC 10234's TSA 24012 and TSB 24014 are each 8 word by 25 bit memories.

Referring to PC 10234's LRUL 24080, a single bit is sufficient to indicate which of the two frames in each of PC 10234's sets was most recently accessed. PC 10234's LRUL 24080's memory is thereby 8 words, or sets long, one bit wide.

As previously described, PC 10234 entries comprise information regarding access rights of certain active subjects to certain active objects. Each PC 10234 entry contains 35 bits of information. Three bits of this information indicate whether a particular subject was read, write, or execute rights relative to a particular object. The remaining 32 bits effectively comprise a length field indicating the volume or portion, that is the number of data bits, of that object to which those access rights pertain.

Referring again to Fig. 240, PC 10234 differs from the generalized cache and from NC 10226 and ATU 10228 in further including Extent Check Logic (EXTCHK) 24082 and Operation Check Logic (OPRCHK) 24084. PC 10234 entries include, as described above, 3 bits identifying type of access rights a particular subject has to a particular object. These 3 bits, representing a Read (R), Write (W), or Execute (E) right, are provided to a

first input of OPRCHK 24084. A second input of OPRCHK 24084 is provided from FUCTL 20214 and specifies whether JP 10114 intends to perform a Read (RI), a Write (WI), or Execute (EI), operation with respect to that object. OPRCHK 24084 compares OPRCHK 24084 access right inputs from DS 24050 to OPRCHK 24084's intended operation input from FUCTL 20214. If that subject does not possess the rights to that object which are required to perform the operation intended by JP 10114, OPRCHK 24084 generates an Operation Violation (OPRV) indicating that a protection violation has occurred.

Similarly, the 32 bits of a PC 10234 entry regarding extent rights is provided as an input (EXTENT) to EXTCHK 24082. As stated above, EXTENT field of PC 10234 entry indicates the length or number of data bits, within an object, to which those access rights pertain. EXTENT field from PC 10234 entry is compared, by EXTCHK 24082, to offset field of the logical descriptor of the current JP 10114 request to MEM 10112 for which a current protection mechanism check is being made. If comparison of extent rights and offset field indicate that the current memory request goes beyond the object length to which the corresponding rights read from DS 24050 apply, EXTCHK 24082 generates an Extent Violation (EXTV) output. EXTV indicates that a current memory request by JP 10114 refers to a portion of an object to which the PC 10234 entry read from BS 24050 does not apply. As described previously, each read from or write to MEM 10112, even as part of a string transfer, is a 32 bit word. As such, EXTCHK 24082 will generate an EXTV output when OFFSET field of a current logical descriptor describes a segment of an object less than 32 bits from the limit defined by EXTENT field of the PC 10234 entry provided in response to that logical descriptor. EXTV and OPRV are gated together, by Protection Violation Gate (PVG) 24086 to generate Protection Violation (PROTV) output indicating that either an extent or an operation violation has occurred.

Having described the structure and operation of MEMINT 20212, and previously the structure and operation of DESP 20210, structure and operation of FUCTL 20214 will be described next below.

3. Fetch Unit Control Logic 20214 (Fig. 202)

The following descriptions will provide a detailed description of FU 10120's structure and operation. Overall operation of FU 10120 will be described first, followed by description of FU 10120's structure, and finally by a detailed description of FU 10120 operation.

As previously described, FUCTL 20214 directs operation of JP 10114 in executing procedures of user's processes. Among the functions performed by FUCTL 20214 are, first, maintenance and operation of CS 10110's Name Space, UID, and AON based addressing system, previously described; second, interpretation of SOPs of user's processes to provide corresponding sequences of microinstructions to FU 10120 and EU 10122 to control operation of JP 10114 in execution of user's processes, previously described; and, third, control of operation of CS 10110's internal mechanisms, for example CS 10110's stack mechanisms.

As will be described in further detail below, FUCTL 20214 includes prefetcher (PREF) 20260 which generates a sequence of logical addresses, each logical address comprising an AON and an offset field, for reading S-Instructions (SINs) of a user's program from MEM 10112. As previously described, each SIN may be comprised of an S-Operation (SOP) and one or more operand Names and may occupy one or more 32 bit words. SINs are read from MEM 10112 as a sequence of single 32 bit words, so that PREF 20260 need not specify a length field in a MEM 10112 read request for an SIN. SINs are read from MEM 10112 through MOD Bus 10144 and are captured and stored in Instruction Buffer (INSTB) 20262. PARSER 20264 extracts, or parses, SOPs and operand Names from INSTB 20262. PARSER 20264 provides operand Names to NC 10226 and SOPs to FUS Interpreter Dispatch Table (FUSDT) 11010 and to EU Dispatch Table (EUSDT) 20266 through Op-Code Register (OPCODEREG) 20268. Operation of INSTB 20262 and PARSER 20264 is controlled by Current program Counter (CPC) 20270, Initial Program Counter (IPC) 20272, and Executed program Counter (EPC) 20274.

As previously described, FUSDT 11010 provides, for each SOP received from OPCODEREG 20268, a corresponding S-Interpreter Dispatch (SD) Pointer, or address, to FUSITT 11012 to select a corresponding sequence of microinstructions to direct operation of JP 10114, in particular FU 10120. As previously described, FUSITT 11012 also contains sequences of microinstructions for controlling and directing operation of CS 10110's internal mechanisms, for example those mechanisms such as RCWS 10358 which are involved in swapping of processes. EUSDT 20266 performs an analogous function with respect to EU 10122 and provides SD Pointers to EU S-Interpreter Tables (EUSITTs) residing in EU 10122.

Micro-Program Counter (mPC) 20276 provides sequential addresses to FUSITT 11012 to select individual microinstructions of sequences of microinstructions. Branch and Case Logic (BRCASE) 20278 provides addresses to FUSITT 11012 to select microinstructions sequences for microinstructions branches and and cases. Repeat Counter (REPCTR) 20280 and Page Number Register (PNREG) 20282 provide addresses to FUSITT 11012 during FUSITT 11012 load operations.

As previously described, FUSITT 11012 is a writable microinstruction control store which is loaded with selected S-Interpreters (SINTs) from MEM 10112.

FUSITT 11012 addresses are also provided by Event Logic (EVENT) 20284 and by JAM input from NC 10226. As will be described further below, EVENT 20284 is part of FUCTL 20214's circuitry primarily concerned with operation of CS 10110's internal mechanisms. Input JAM from NC 10226 initiates certain FUCTL 20214 control functions for CS 10110's Name Space addressing mechanisms, and in particular NC 10226. Selection between the above discussed address inputs to FUSITT 11012 is controlled by S-

Interpreter Table Next Address Generator Logic (SITTNAG) 20286.

Other portions of FUCTL 20214's circuitry are concerned with operation of CS 10110's internal mechanisms. For example, FUCTL 20214 includes Return Control Word Stack (RCWS) 10358, previously described with reference to CS 10110's Stack Mechanisms. Register Address Generator (RAG) 20288 provides pointers for addressing of GRF 10354 and RCWS 10358 and includes Micro-Stack Pointer Registers (MISPR) 10356.

As previously described, MISPR 10356 mechanism provides pointers for addressing Micro-Stack (MIS) 10368. As will be described further below, actual MIS 10368 Pointers pointing to current, previous, and bottom frames of MIS 10368 reside in Micro-Control Word Register 1 (MCW1) 20290. MCW1 20290 and Micro-Control Word Zero Register (MCWO) 20292 together contain certain information indicating the current execution environment of a microinstruction sequence currently being executed by FU 10120. This execution information is used in aid of execution of these microinstruction sequences. State Registers (STATE) 20294 capture and store certain information regarding state of operation of FU 10120. As described further below, this information, referred to as state vectors, is used to enable and direct operation of FU 10120.

Timers (TIMERS) 20296 monitor elapsed time since occurrence of the events requiring servicing by FU 10120. If waiting time for these events exceeds certain limits, TIMERS 20296 indicate that these limits have been exceeded so that service of those events may be initiated.

Finally, Fetch Unit to E Unit Interface Logic (FUEUINT) 20298 comprises the FU 10120 portion of the interface between FU 10120 and EU 10122. FUEUINT 20298 is primary path through which operation of FU 10120 and EU 10122 is coordinated.

Having described overall operation of FU 10120, structure of FU 10120 will be described next below with aid of Fig. 202, description of FU 10120's structure will be followed by a detailed description of FU 10120 wherein further, more detailed, diagrams of certain portions of FU 10120 will be introduced as required to enhance clarity of presentation.

a.a. Fetch Unit Control Logic 20214 Overall Structure

Referring again to Fig. 202, as previously described Fig. 202 includes a partial block diagram of FUCTL 20214. Following the same sequence of description as above, PREF 20260 has a 28 bit bi-directional port connected to AON Bus 20230 and 32 bit bi-directional port directed from OFFSET Bus 20228. A control input of PREF 20260 is connected from control output of INSTB 20262.

INSTB 20262 32 bit data input (DI) is connected from MOD Bus 10144. INSTB 20262's 16 bit output (DO) is connected to 16 bit bi-directional input of OPCODEREG 20268 and to 16 bit NAME Bus 20224. OPCODEREG 20268's input comprises 8 bits of SINT and 3 bits of dialect selection. As previously described, NAME Bus 20224 is connected to 16 bit bi-directional port of Name Trap (NT) 20254, to address input ADR of NC 10226, and to inputs and outputs of OFFP 20228. Control inputs of INSTB 20262 and PARSER 20264 are connected from a control output of CPC 20270.

Thirty-two bit input of CPC 20270 is connected from JPD Bus 10142 and CPC 20270's 32 bit output is connected to 32 bit input of IPC 20272. Thirty-two bit output of IPC 20272 is connected to 32 bit input of EPC 20274 and to JPD Bus 10142. EPC 20274's 32 bit output is similarly connected to JPD Bus 10142.

Eleven bit outputs of OPCODEREG 20268 are connected to 11 bit address inputs of FUSDT 11010 and EUSDT 20266. These 11 bit address inputs to FUSDT 11010 and EUSDT 20266 each comprise 3 bits of dialect selection code and 8 bits of SINT code. Twelve bit SDT outputs of EUSDT 20266 is connected to inputs of Microinstruction Control Store in EU 10122, as will be described in a following description of EU 10122. FUSDT 11010 has, as described further below, two outputs connected to address (ADR) Bus 20298. First output of FUSDT 11010 are six bit SDT pointers, or addresses, corresponding to generic SINTs as will be described further below. Second output of FUSDT 11010 are 15 bit SDT pointers, or addresses, for algorithm microinstruction sequences, again as will be described further below.

Referring to RCWS 10358, RCWS 10358 has a first bidirectional port connected from JPD Bus 10142. Second, third, and fourth bi-directional ports of RCWS 10358 are connected from, respectively, a bi-directional port of MCW1 20290, a first bi-directional port EVENT 20284, and a bi-directional port of mPC 20276. An output of RCWS 10358 is connected to ADR Bus 20298.

An input of mPC 20276 is connected from ADR Bus 20298 and first and second outputs of mPC 20276 are connected to, respectively, an input of BRCASE 20278 and to ADR Bus 20298. An output of BRCASE 20278 is connected to ADR Bus 20298.

As described above, a first bi-directional port of EVENT 20284 is connected to RCWS 10358. A second bidirectional port of EVENT 20284 is connected from MCWO 20292. An output of EVENT 20284 is connected to ADR Bus 20298.

Inputs of RPCTR 20280 and PNREG 20282 are connected from JPD Bus 10142. Outputs of RPCTR 20280 and PNREG 20282 are connected to ADR Bus 20298.

ADR Bus 20298, and an input from a first output of FUSITT 11012, are connected to inputs of SITTNAG 20286.

Output of SITTNAG 20286 is connected, through Control Store Address (CSADR) Bus 20299, to address input of FUSITT 11012. Data input of FUSITT 11012 is connected from JPD Bus 10142. Control outputs of FUSITT 11012 are connected to almost all elements of JP 10114 and thus, for clarity of presentation, are not

shown in detail by drawn physical connections but are described in following descriptions.

As described above, MCW0 20292 and MCW1 20290 have bi-directional ports connected to, respectively, bidirectional ports of EVENT 20284 and to a second bidirectional port of RCWS 10358. Outputs of MCW0 20292 and MCW1 20290 are connected to JPD Bus 10142. Other inputs of MCW0 20292 and MCW1 20290, as will be described further below, are connected from several other elements of JP 10114 and, for clarity of presentation, are not shown herein in detail but are described in the following text. STATE 20294 similarly has a large number of inputs and outputs connected from and to other elements of JP 10114, and in particular FU 10120. Inputs and outputs of STATE 20294 are not indicated here for clarity of presentation and will be described in detail below.

RAG 20288 has an input connected from JPD Bus 10142 and other inputs connected, for example, from MCW1 20290. RAG 20288, including MISPR 10356, provides outputs, for example, as address inputs to RCWS 10358 and GRF 10354. Again, for clarity of presentation, inputs and outputs of RAG 20288 are not shown in detail in Fig. 202 but will be described in detail further below.

TIMERS 20296 receive inputs from EVENT 20284 and FUSITT 11012 and provide outputs to EVENT 20284. For clarity of presentation, these indications are not shown in detail in Fig. 202 but will be described further below.

FUJNT 20298 receives control inputs from FUSITT 11012 and EU 10122. FUJNT 20298 provides outputs to EU 10122 and to other elements of FUJCTL 20214. For clarity of presentation, connections to and from FUJNT 20298 are not shown in detail in Fig. 202 but will be described in further detail below.

Having described the overall operation, and structure, of FUJCTL 20214, operation of FUJCTL 20214 will be described next below. During the following descriptions further diagrams of certain portions of FUJCTL 20214 will be introduced as required to disclose structure and operation of FUJCTL 20214 to one of ordinary skill in the art. FUJCTL 20214's operation with regard to fetching and interpretation of SINS, that is SOPs and operand Names, will be described first, followed by description of FUJCTL 20214's operation with regard to CS 10110's internal mechanisms.

b.b. Fetch Unit Control Logic 20214 Operations

Referring first to those elements of FUJCTL 20214 directly concerned with control of JP 10114 in response to SOPs and Name syllables, those elements include: (1) PREF 20260; (2) INSTB 20262; (3) PARSER 20264; (4) CPC 20270, IPC 20272, and EPC 20274; (5) OPCODEREG 20268; (6) FUSDT 11010 and EUSDT 20266; (7) mPC 20276; (8) BRCASE 20278; (9) REPCTR 20280 and PNREG 20282; (10) a part of RCWS 10358; (11) SITTNAG 20286; (12) FUSITT 11012; and, (13) NT 20254. These FUJCTL 20214 elements will be described below in the order named.

a.a.a. Prefetcher 20260, Instruction Buffer 20262, Parser 20264, Operation Code Register 20268, CPC 20270, IPC 20272, and EPC 20274 (Fig. 241)

As described above, PREF 20260 generates a series of addresses to MEM 10112 to read SINS of user's programs from MEM 10112 to FUJCTL 20214, and in particular to INSTB 20262. Each PREF 20260 read request transfers one 32 bit word from MEM 10112. Each SIN may be comprised of an SOP and one or more Name syllables. Each SOP may comprise, for example, 8 bits of information while each Name syllable may comprise, for example, 8, 12, or 16 bits of data. In general, and as will be described in further detail in a following description of STATE 20294, PREF 20260 obtains access to MEM 10112 on alternate 110 nanosecond system clock cycles. PREF 20260's access to MEM 10112 is conditional upon INSTB 20262 indicating that INSTB 20262 is ready to receive an SIN read from MEM 10112. In particular, INSTB 20262 generates control output Quiry Prefetch (QPF) to PREF 20260 to enable PREF 20260 to submit a request to MEM 10112 when, as described further below, INSTB 20262 is ready to receive an SIN read from MEM 10112.

PREF 20260 is a counter register comprised, for example of SN74S163s.

Bi-directional inputs and outputs of PREF 20260 are connected to AON Bus 20230 and OFFSET Bus 20228. As PREF 20260 reads only single 32 bit words, PREF 20260 is not required to specify a LENGTH field as part of an SIN read request, that is an AON and an OFFSET field are sufficient to define a single 32 bit word. At start of read of a sequence of SINS from MEM 10112, address (AON and OFFSET fields) of first 32 bit word of that SIN sequence are provided to MEM 10112 by DESP 20210 and concurrently loaded, from AON Bus 20230 and OFFSET Bus 20228, into PREF 20260. Thereafter, as each successive thirty-two bit word of the SIN's sequence is read from MEM 10112, the address residing in PREF 20260 is incremented to specify successive 32 bit words of that SIN's sequence. The successive single word addresses are, for all words after first word of a sequence, provided to MEM 10112 from PREF 20260.

As described above, INSTB 20262 receives SINS from MEM 10112 through MOD Bus 10144 and, with PARSER 20264 and operating under control of CPC 20270, provides Name syllables to NAME Bus 20224 and SINS to OPCODEREG 20268. INSTB 20262 is provided, together with PREF 20260 to increase execution speed of SINS.

Referring to Fig. 241, a more detailed block diagram of INSTB 20262, PARSER 20264, CPC 20270, IPC 20272, EPC 20274 as shown. INSTB 20262 is shown as comprising two 32 bit registers having parallel 32 bit inputs from MOD Bus 10144. INSTB 20262 also receives two Write Clock (WC) inputs, one for each 32 bit register of INSTB 20262, from Instruction Buffer Write Control (INSTBWC) 24110. INSTB 20262's outputs

are structured as eight, eight bit Basic Syllables (BSs), indicated as BSO to BS7. BSO, BS2, BS4, and BS6 are ORed to comprise eight bit Basic Syllable, Even (BSE) of INSTB 20262 while BSO, BS3, BS5, and BS7 are similarly ORed to comprise Basic Syllable, Odd (BSO) of INSTB 20262. BSO and BSE are provided as inputs of PARSER 20264.

5 PARSER 20264 receives a first control input from Current Syllable Size Register (CSSR) 24112, associated with CPC 20270. A second control input of PARSER 20264 is provided from Instruction Buffer Syllable Decode Register (IBSDECR) 24114, also associated with CPC 20270. PARSER 20264 provides an eight bit output to NAME Bus 20224 and to input of OPCODEREG 20268.

10 Referring to INSTBWC 24110, INSTBWC 24110 provides, as described further below, control signals pertaining to writing of SInS into INSTB 20262 from MOD Bus 10144. INSTBWC 24110 also provides control signals pertaining to operation of PREF 20260. In addition to WC outputs to INSTB 20262, INSTBWC 24110 provides control output QPF to PREF 20260, control output Instruction Buffer Hung (IBHUNG) to EVENT 20284, and control signal Instruction Buffer Wait (IBWAIT) to STATE 20294. INSTBWC 24110 also receives a control input BRANCH from BRCASE 20278 and an error input from TIMERS 20296.

15 Referring to CPC 20270, IPC 20272, and EPC 20274, IPC 20272 and EPC 20274 are represented in Fig. 241 as in Fig. 202. Further FUCTL 20214 circuitry is shown as associated with CPC 20270. CPC 20270 is a twenty-nine bit register receiving bits one to twenty-five (CPC(1—25)) from bits one to twenty-five of JPD Bus 10142. CPC 20270 Bit 0 (CPC0) is provided from CPC0 CPC0 Select (CPCOS) 24116. Inputs of CPCOS 24116 are Bit 1 output from CPC 20270 (CPC1) and Bit 0 from JPD Bus 10142. Bits twenty-six, twenty-seven, and twenty-eight of CPC 20270 (CPC(2628)) are provided from CPC Multiplexer (CPCMUX) 24118. CPCMUX 24118 also provides an input to IBSDECR 24114. Inputs of CPCMUX 24118 are bits twenty-five, twenty-six, and twenty-eight from JPD Bus 10142 and a three bit output of CPC Arithmetic and Logic Unit (CPCALU) 24120. A first input of CPCALU 24120 is connected from output bits 26, 27, and 28 of CPC 20270. Second input of CPCALU 24120 is connected from CSSR 24112. CSSR 24112's input is connected from JPD Bus 10142.

25 As described above, INSTB 20262 is implemented as a sixty-four bit wide register. INSTB 20262 is organized as two thirty-two bit words, referred to as Instruction Buffer Word 0 (IB0) and Instruction Buffer Word 1 (IB1), and operates as a two word, first-in-first-out buffer memory. PREF 20260 loads one of IB0 or IB1 on each memory reference by PREF 20260. Only PREF 20260 may load INSTB 20262, and INSTB 20262 may be loaded only from MOD Bus 10144. Separate clocks, respectively Instruction Buffer Write Clock 0 (IBWC0) and Instruction Buffer Write Clock 1 (IBWC1), are provided from INSTBWC 24110 to load, respectively, IBW0 and IBW1 into INSTB 20262. IBWC0 and IBWC1 are each a gated 110 nano-second clock. An IBW0 or an IBW1 is written into INSTB 20262 when, respectively, IBWC0 or IBWC1 is enabled by INSTBWC 24110. IBWC0 and IBWC1 will be enabled only when MEM 10112 indicates that data for INSTB 20262 is available by asserting interface control signal DAVI as previously discussed.

30 INSTBWC 24110 is primarily concerned with control of FU 10120 with respect to writing of SInS into INSTB 20262. As described above, INSTBWC 24110 provides IBWC0 and IBWC1 to INSTB 20262. IBWC0 and IBWC1 are enabled by INSTBWC 24110's input DAVI from MEM 10112. Selection between IBWC0 and IBWC1 is controlled by INSTBWC 24110's input from CPC 20270. In particular, and as will be described further below, Bit 26 (CPC 26) of CPC 20270's twenty-nine bit word indicates whether IBW0 or IBW1 is written into INSTB 20262.

40 In addition to controlling writing of IBW0 and IBW1 into INSTB 20262, INSTBWC 24110 provides control signals to elements of FU 10120 to control reading of SInS from MEM 10112 to INSTB 20262. In this regard, INSTBWC 24110 detects certain conditions regarding status of SIN words in INSTB 20262 and provides corresponding control signals, described momentarily, to other elements of FU 10120 so that INSTB 20262 would generally always contain at least one valid SOP or Name syllable. First, if INSTB 20262 is not full, that is either IBW0 or IBW1 or both is invalid, for example because IBW0 has been read from INSTB 20262 and executed, INSTBWC 24110 detects this condition and provides control signal QPF to PREF 20262 to initiate a read from MEM 10112. INSTBWC 24110 currently enables either IBW0 or IBW1 portion of INSTB 20262 to receive the word read from MEM 10112 in response to PREF 20260's request. As stated above, this operation will be initiated when INSTBWC 24110 detects and indicates, by generating a validity flag, that either IBW0 or IBW1 is invalid. In this case, IBW0 or IBW1 will be indicated as invalid when read from INSTB 20262 by PARSER 20264. As will be described further below, INSTBWC 24110 validity flags for IBW0 and IBW1 are generated by INSTBWC 24110 control inputs comprising Bits 26 to 28 (CPC 26—28) from CPC 20270 and by current syllable size or value, flag (K) input from CSSR 24112. Secondly, INSTBWC 24110 will detect when INSTB 20262 is empty, that is when both IBW0 and IBW1 are invalid, as just described, or when only a half of a sixteen bit Name syllable is present in INSTB 20262. In response to either condition, INSTBWC 24110 will generate control signal IBWAIT to STATE 20294. As will be described further below, IBWAIT will result in suspension of execution of microinstructions referencing INSTB 20262. PREF 20260 requests to MEM 10112 will already have been initiated, as described above unless certain other conditions, described momentarily, occur. Thirdly, INSTBWC 24110 will detect when INSTB 20262 is empty and PREF 20262 is hung, that is unable to submit requests to MEM 10112, and a current microinstruction is attempting to parse a syllable from INSTB 20262. In this case, INSTBWC 24110 will generate control signal Instruction Buffer Hung (IBHUNG) to EVENT 20284. As will be described further below, IBHUNG will result in initiation of a microinstruction sequence to restore flow of words to INSTB 20262. Fourthly, INSTBWC 24110 will detect, through microinstruction control signals provided from FUSITT 11012, when a branch in

a microinstruction sequence provided by FUSITT 11012 in response to an SOP occurs. In this case, both IBW0 and IBW1 will be flagged as invalid. INSTBWC 24110 will then ignore SIN words being read from MEM 10112 in response to a previously submitted PREF 20260 request, but not yet received at the time the branch occurs. This prevents INSTB 20260 from receiving invalid SIN words; PREF 20260 and INSTB 20262 will then proceed to request and receive valid SIN words of the branch.

As described above, PARSER 20264, operating under control of CPC 20270 and CPC 20270 associated circuitry, reads Name syllables and SOPs from INSTB 20262 to, respectively, NAME Bus 20224 and OPCODEREG 20268. PARSER 20264 operates as a multiplexer with associated control logic.

As previously described, INSTB 20262 is internally structured as eight, eight bit words, BS0 to BS7. IBW0 comprises BS0 to B3 while IBW1 comprises BS4 to BS7. Each SOP is comprised of eight bits of data and thus comprises one Basic Syllable while each Name syllable comprises 8, 12, or 16 bits of data and thus comprises either one or two Basic Syllables. Name syllable size, as previously stated, is indicated by Current Syllable Size Value K stored in CSSR 24112.

BS0 and BS4 are loaded into INSTB 20262 from MOD Bus 10144 bits zero to seven while BS2 and BS6 are loaded from MOD Bus 10144 bits sixteen to twenty-three. BS1 and BS5 are loaded from MOD Bus 10144 bits eight to fifteen while BS3 and BS7 are loaded from MOD Bus 10144 bits twenty-four to thirty-one. Odd numbered Basic Syllable outputs BS1, BS3, BS5, and BS7 are ORed to comprise eight bit Basic Syllable, Odd output BS0 of INSTB 20262. Even numbered Basic Syllable outputs BS0, BS2, BS4 and BS6 of INSTB 20262 are similarly ORed to comprise eight bit Basic Syllable, Even output BSE. At any time, one odd numbered Basic Syllable output and one even numbered Basic Syllable output of INSTB 20262 are selected as inputs to PARSER 20264 by Instruction Buffer Read Enable (IBORE) enable and selection signals provided to INSTB 20262 by IBSDECR 24114. IBSDECR 24114 includes decoding circuitry. Input to IBSDECR 24114's decoding logic is comprised of three bits (RCPC(26—28)) provided from CPCMUX 24118. As indicated in Fig. 241, CPC (26—28) may be provided from JPD Bus 10142 bits 25 to 28 or from output of CPCALU 24120. One input CPCALU 24120 is CPC (26—28) from CPC 20270. Operation of CPC 20270 and CPC 20270's associated circuitry will be described further below. RCPC (26—28) is decoded by IBSDECR 24114 to generate IBORE (0—7) to INSTB 20262. RCPC 26 and RCPC 27 are decoded to select one of the four odd numbered Basic Syllable outputs (that is BS1, BS3, BS5 or BS7) of INSTB 20262 as the odd numbered basic syllable input to PARSER 20264. RCPC 28 selects either the preceding or the following even numbered Basic Syllable output of INSTB 20262 as the even numbered Basic Syllable input to PARSER 20264. The eight decoded bits of IBORE (0—7) generated by IBSDECR 24114 decoding logic are loaded into IBSDECR 24114 eight bit register and subsequently provided to INSTB 20262 as IBORE (0—7).

PARSER 20264 selects BS0, or BSE, or both BS0 and BSE, as PARSER 20264's output to NAME Bus 20224 or to OPCODEREG 20268. In the case of an SOP or an eight bit Name syllable, either BS0 or BSE will be selected as PARSER 20264's output. In the case of a twelve or sixteen bit Name syllable, both BS0 and BSE may be selected as PARSER 20264's output. PARSER 20264 operation is controlled by microinstruction control outputs from FUSITT 11012.

Program counters IPC 20272, EPC 20274, and CPC 20270 are associated with control of fetching and parsing of SInS. In general, IPC 20272, EPC 20274, and CPC 20270 operate under microinstruction control from FUSITT 11012.

CPC 20270 is Current Program Counter and contains 28 bits pointing to the current syllable in INSTB 20272. Bits 29 to 31 of CPC 20270 are not provided, so the bits 29 to 31 of CPC 20270's output are zero, which guarantees byte boundaries for SOPs. Contents of CPC 20270 are thereby also a pointer which is a byte align offset into a current procedure object. Initial Program Counter (IPC) 20272 is a buffer register connected from output of CPC 20270 and provided for timing overlap. IPC 20272 may be loaded only from CPC 20270 which, as previously described, is 29 bits wide, that does not contain bits 29, 30, and 31 which are forced to zero in IPC 20272. IPC 20272 may be read onto JPD Bus 10142 as a start value in an unconditional branch.

EPC 20274 is a thirty-two bit register usually containing a pointer to the current SOP being executed. Upon occurrence of an SOP branch, the pointer in EPC 20274 will point to the SOP from which the branch was executed. The pointer residing in EPC 20274 is an offset into a current procedure object. EPC 20274 may be loaded only from IPC 20272, and may be read onto JPD Bus 10142.

Referring again to CPC 20270, as described above CPC 20270 is a current syllable counter. CPC 20270 contains a pointer to the next SOP syllable, or Base Syllable, to be parsed by PARSER 20264. As SOPs are always on byte boundaries, CPC 20270 pointer is 29 bits wide, CPC (0—28). The three low order bits of CPC 20270's pointer, that is CPC (29—31), do not physically exist and are assumed to be always zero. CPC 20270's pointer to next instruction syllable to be parsed thereby always points to byte boundaries.

CPC 20270 bits 26 to 28, CPC (26—28), indicate, as described above, a particular Base Syllable in INSTB 20262. Bits 0—25 (CPC(0—25)) of CPC 20270 indicate 32 bit words, read into INSTB 20262 as IBW0 and IBW1, of a sequence of SInS. CPC 20270 pointer is updated each time a parse operation reading a Base Syllable from INSTB 20262 is executed. As previously described, these parsing operations are performed under microinstruction control from FUSITT 11012.

Conceptually, CPC 20270 is organized as a twenty-six bit counter, containing CPC (0—25), with a three bit register appended on the low order side, as CPC (26—28). This organization is used because CPC (26—28) counts INSTB 20262 Base Syllables parsed and must be incremented dependant upon current

Name Syllable Size K stored CSSR 24112. CPC (0—25), however, counts successive thirty-two bit words of a sequence of SInS and may thereby be implemented as a binary counter. As shown in Fig. 241, CPC (26—28) is loaded from output of CPCMUX 24118. A first input of CPCMUX 24118 is connected from bits 29 to 31 of JPD Bus 10142. This input to CPC (26—28) from JPD Bus 10142 is provided to allow CPC 20270 to be loaded from JPD Bus 10142, for example when loading CPC 20270 with an initial pointer value. Second input of CPCMUX 24118 is from output of CPCALU 24120 and is the path by which CPC (26—28) is incremented as successive Base Syllables are parsed from INSTB 20262. A first input of CPCALU 24120 is CPC (26—28) from CPC 20270. Second input of CPCALU 24120 is a dual input from CSSR 24112. First input from CSSR 24112 is logic 1 in the least significant bit position, that is in position corresponding to CPC (28). This input is used when single Base Syllables are parsed from INSTB 20262, for example in an eight bit SOP or an eight bit Name syllable. CSSR 24112's first input to CPCALU 24120 increments CPC (0—32) by eight, that is one to CPC (26—28), each time a single Base Syllable is parsed from INSTB 20262. Second input to CPCALU 24120 from CSSR 24112 is K, that is current Name Syllable size. As previously described, K may be eight, twelve, or sixteen. CPC (26—28) is thereby incremented by one when K equals eight and is incremented by two when K equals twelve or sixteen. As shown in Fig. 241, K is loaded into CSSR 24112 from JPD Bus 10142.

CPC (0—25), as described above, operates as a twenty-six bit counter which is incremented each time CPC (26—28) overflows. CPC (0—25) is incremented by carry output of CPCALU 24120. In actual implementation, CPC 20270 is organized to reduce the number of integrated circuits required. CPC (1—25) is constructed as a counter and inputs of CPC (1—25) counter are connected from bits 1 to 24 of JPD Bus 10142 to allow loading of an initial value of CPC 20270 pointer. CPC (0) and CPC (26—28) are implemented as a four bit register. Operation of CPC (26—28) portions of this register have been described above. Input of CPC (0) portion of this register is connected from output of CPCOS 24116. CPCOS 24116 is a multiplexer having a first input connected from bit 0 of JPD Bus 10142. This input from JPD Bus 10142 is used, for example, when loading CPC 20272 with an initial pointer value. Second input of CPCOS 24116 is overflow output of CPC (1—25) counter and allows CPC (0) portion of the four bit register and CPC (1—25) counter to operate as a twenty-six bit counter.

Finally, as shown in Fig 241, output of CPC 20270 may be loaded into IPC 20272. An initial CPC 20270 pointer value may therefore be written into CPC 20270 from JPD Bus 10142 and subsequently copied into IPC 20272.

Referring again to PARSER 20264, as described above PARSER 20264 reads, or parses, basic syllables from INSTB 20262 to NAME Bus 20224. Input of PARSER 20264 is a sixteen bit word comprised of an eight bit odd numbered Base Syllable, BSO, and an eight bit even numbered Base Syllable, BSE. Depending upon whether PARSER 20264 is parsing an eight bit SOP, an eight bit Name syllable, a twelve bit Name syllable, or sixteen bit Name syllable, PARSER 20264 may select BSO, BSE, or both BSO and BSE, as output onto NAME Bus 20224.

If PARSER 20264 is parsing Name syllables and K is not equal to eight, that is equal to twelve or sixteen, PARSER 20264 transfers both BSO and BSE onto NAME Bus 20224 and determines which of BSO or BSE is most significant. The decision as to whether BSO or BSE is most significant is determined by CPC (28). If CPC (28) indicates BSO is most significant, BSO is transferred onto NAME Bus 20224 bits 0 to 7 (NAME(0—7)) and BSE onto NAME Bus 20224 bits eight to fifteen (NAME(8—15)). If CPC (28) indicates BSE is most significant, BSE is transferred onto NAME (0—7) and BSO onto NAME (8—15). This operation insures that Name syllables are parsed onto NAME Bus 20224 in the order in which occur in the SIN stream.

If PARSER 20264 is parsing Name syllables of Syllable Size K = 8, PARSER 20264 will select either BSO or BSE, as indicated by CPC (28), as output to NAME (0—7). PARSER 20264 will place 0's on NAME (8—15).

If PARSER 20264 is parsing SOPs of eight bits, PARSER 20264 will select BSO or BSE as output to NAME (0—7) as selected by CPC (28). PARSER 20264 will place 0's onto NAME (8—15). Concurrently, PARSER 20264 will generate OPREG to OPCODEREG 20268 to enable transfer of NAME (0—7) into OPCODEREG 20268. OPCODEREG 20268 is not loaded when PARSER 20264 is parsing Name syllables. The microinstruction input from FUSITT 11012 which controls PARSER 20264 operation also determines whether PARSER 20264 is parsing an SOP or a Name syllable and controls generation of OPREG.

Operation of NC 10226, which receives Name syllables as address inputs from NAME Bus 20224, has been discussed previously with reference to MEMINT 20212. Name Trap (NT) 20254 is connected from NAME Bus 20224 to receive and capture Name syllables parsed onto NAME Bus 20224 by PARSER 20264. Operation of NT 20254 has been also previously discussed with reference to MEMINT.

b.b.b. Fetch Unit Dispatch Table 11010, Execute Unit Dispatch Table 20266 and Operation Code Register 20268 (Fig. 242)

As previously described, CS 10110 is a multiple language machine. Each program written in a high level user language is compiled into a corresponding S-Language program containing S-Language Instructions referred to as SOPs. CS 10110 provides a set or dialect, of microcode instructions, referred to as S-Interpreters (SINTs) for each S-Language. SINTs interpret SOPs to provide corresponding sequences of microinstructions for detailed control of CS 10110 operations. CS 10110's SINTs for FU 10120 and EU 10122 operations are stored, respectively, in FUSITT 11012 and in a corresponding control store memory in EU 10122, described in a following description of EU 10122. Each SINT comprises one or more sequences

of microinstructions, each sequence of microinstructions corresponding to a particular SOP in a particular S-Language dialect. Fetch Unit S-Interpreter Dispatch Table (FUSDT) 11010 and Execute Unit S-Interpreter Dispatch Table (EUSDT) 20266 contain an S-Interpreter Dispatcher (SD) for each S-Language dialect. Each SD is comprised of a set of SD Pointers (SDPs) wherein each SDP in a particular SD corresponds to a particular SOP of that SD dialect. Each SDP is an address pointing to a location, in FUSITT 11012 or EUSITT, of the start of the corresponding sequence of microinstructions for interpreting the SOP corresponding to that SDP. As will be described further below, SOPs received and stored in OPCODEREG 20268 are used to generate addresses into FUSDT 11010 and EUSDT 20266 to select corresponding SDPs. Those SDPs are then provided to FUSITT 11012 through ADR 20202, or to EUSITT through EUDIS Bus 20206, to select corresponding sequences of microinstructions from FUSITT 11012 and EUSITT.

Referring to Fig. 242, a more detailed block diagram of OPCODEREG 20268, FUSDT 11010, and EUSDT 20266 is shown. As shown therein, OPCODEREG 20268 is comprised of OP-Code Latch (LOPCODE) 24210, Dialect Register (RDIAL) 24212, Load Address Register (LADDR) 24214, and Fetch Unit Dispatch Encoder (FUDISENC) 24216. Data inputs of LOPCODE 24210 are connected from NAME Bus 20224 to receive SOPs parsed from INSTB 20262. Load inputs of RDIAL 24212 are connected from Bits 28 to 31 of JPD Bus 10142. Outputs of LOPCODE 24210, RDIAL 24212 and LADDR 24214 are connected to inputs of FUDISENC 24216. Outputs of FUDISENC 24216 are connected to address inputs of FUSDT 11010 and EUSDT 20266.

FUSDT 11010 is comprised of Fetch Unit Dispatch File (FUDISF) 24218 and Algorithm File (AF) 24220. Address inputs of FUDISF 24218 and AF 24220 are connected, as previously described, from address outputs of FUDISENC 24216. Data load inputs of FUDISF 24218 and AF 24220 are connected from, respectively, Bits 10 to 15 and Bits 16 to 31 of JPD Bus 10142. SDP outputs of FUDISF 24218 and AF 24220 are connected to ADR Buses 20202.

EUSDT 20266 is comprised of Execute Unit Dispatch File (EUDISF) 24222 and Execute Unit Dispatch Selector (EUDISS) 24224. Address inputs of EUDISF 24222 are, as described above, connected from outputs of FUDISENC 24216. Data load inputs of EUDISF 24222 are connected from Bits 20 to 31 of JPD Bus 10142. Inputs of EUDISS 24224 are connected from SDP output of EUDISF 24222, from Bits 20 to 31 of JPD Bus 10142, and from Microcode Literal (mLT) output of FUSITT 11012. SDP outputs of EUDISS 24224 are connected to EUDIS Bus 20206.

As previously described, OPCODEREG 20268 provides addresses, generated from SOPs loaded into OPCODEREG 20268, to FUSDT 11010 and EUSDT 20266 to select SDPs to be provided as address inputs to FUSITT 11012 and EUSITT. LOPCODE 24210 receives and stores eight bit SOPs parsed from INSTB 20262 as described above. OPCODEREG 20268 also provides addresses to FUSDT 11010 and EUSDT 20266 to load FUSDT 11010 and EUSDT 20266 with SDs for S-Language dialects currently being utilized by CS 10110. LOPCODE 24210 and RDIAL 24212, as described below, provide addresses to FUSDT 11010 and EUSDT 20266 when translating SOPs to SDPs and LADDR 24214 provides addresses when FUSDT 11010 and EUSDT 20266 are being loaded with SDs.

Referring first to LADDR 24214, LADDR 24214 has an eight bit counter. Addresses are provided to FUSDT 11010 and EUSDT 20266 from LADDR 24214 only when FUSDT 11010 and EUSDT 20266 are being loaded with SDs, that is groups of SDPs for S-Language dialects currently being utilized by CS 10110. During this operation, output of LADDR 24214 is enabled to FUSDT 11010 and EUSDT 20266 by microcode control signals (not shown for clarity of presentation) from FUSITT 11012. Selection between FUDISF 24218, AF 24220, and EUDISF 24222 to receive addresses is similarly provided by microinstruction enable signals (also not shown for clarity of presentation) provided from FUSITT 11012. These FUSDT 11010 and EUSDT 20266 address enable inputs may select, at any time, any or all of FUDISF 24218, AF 24220, or EUDISF 24222 to receive address inputs. SDPs to be loaded into FUDISF 24218, AF 24220, and EUDISF 24222 are provided, respectively, from Bits 10 to 15 (JPD(10—15)), Bits 16 to 31 (JPD(16—31)), and Bits 20 to 31 (JPD(20—31)) of JPD Bus 10142. Address contents of LADDR 24214 are successively incremented by one as successive SDPs are loaded into FUSDT 11010 and EUSDT 20266. Incrementing of LADDR 24214 is, again, controlled by microinstruction control inputs from FUSITT 11012.

Address inputs to FUSDT 11010 and EUSDT 20266 during interpretation of SOPs are provided from LOPCODE 24210 and RDIAL 24212. LOPCODE 24210 is a register counter having, as described above, data inputs connected from NAME Bus 20224 to receive SOPs from PARSER 20264. In a first mode, LOPCODE 24210 may operate as a latch, loaded with one SOP at a time from output of PARSER 20264. In a second mode, LOPCODE 24210 operates as a clock register to receive successive eight bit inputs from low order eight bits of NAME Bus 20224 (NAME(8—15)). Loading of LOPCODE 24210 is controlled by microinstruction control outputs (not shown for clarity of presentation) from FUSITT 11012.

As will be described further below, eight bit SOPs stored in LOPCODE 24210 are concatenated with the output of RDIAL 24212 to provide addresses to FUSDT 11010 and EUSDT 20266 to select SDPs corresponding to particular SOPs. That portion of these addresses provided from LOPCODE 24210, that is the eight bit SOPs, selects particular SDPs within a particular SD. Particular SDs are selected by that portion of these addresses which is provided from the contents of RDIAL 24212.

RDIAL 24212 receives and stores four bit Dialect Codes indicating the particular S-Language dialect currently being used by CS 10110 and executing the SOPs of a user's program. These four bit Dialect Codes are provided from JPD Bus 10142, as JPD (28—31). Loading of RDIAL 24212 with four bit Dialect Codes is controlled by microinstruction control signals provided from FUSITT 11012 (not shown for clarity of

presentation).

Four bit Dialect Codes in RDIAL 24212 define partitions in FUDISF 24218, AF 24220 and EUDISF 24222. Each partition contains SDPs for a different S-Language dialect, that is contains a different SD. FUDISF 24218, AF 24220 and EUDISF 24222 may contain, for example, eight 128 word partitions or four 256 word partitions. A single bit of Dialect Code, for example Bit 3, defines whether FUDISF 24218, AF 24220, and EUDISF 24222 contain four or eight partitions. If FUSDT 11010 and EUSDT 20266 contain four partitions, the two most significant bits of address into FUSDT 11010 and EUSDT 20266 are provided from Dialect Code Bits 1 and 2 and determine which partition is addressed. The lower order eight bits of address are provided from LOPCODE 24210 and determine which word in a selected partition is addressed. If FUSDT 11010 and EUSDT 20266 contain eight partitions, the three most significant bits of address into FUSDT 11010 and EUSDT 20266 are provided from Bits 0 to 2 of Dialect Code, to select a particular partition, and the lower seven bits of address are provided from LOPCODE 24210 to select a particular word in the selected partition.

As described above, LOPCODE 24210 eight bit output and RDIAL 24212's four bit output are concatenated together, through FUDISENC 24216, to provide a ten bit address input to FUSDT 11010 and EUSDT 20266. FUDISENC 24216 is an encoding circuit and will be described further below with reference to FUDISF 24218. As previously described, selection of FUDISF 24218, AF 24220, and EUDISF 24222 to receive address inputs from RDIAL 24212 and LOPCODE 24210 is controlled by microinstruction control enable inputs provided from FUSITT 11012 (not shown for clarity of presentation).

Referring to FUSDT 11010, both FUDISF 24218 and AF 24220 provide SDPs to FUSITT 11012, but do so for differing purposes. In general, microinstruction control operations may be regarded as falling into two classes. First, there are those microinstruction operations which are generic, that is general in nature and used by or applying to a broad variety of SOPs of a particular dialect or even of many dialects. An example of this class of microinstruction operation is fetches of operand values. FUDISF 24218 provides SDPs for this class of microinstruction operations. As described below, FUDISF 24218 is a fast access memory allowing a single microinstruction control output of FUSITT 11012 to parse an SOP from INSTB 20262 into LOPCODE 24210, and a corresponding SDP to be provided from FUDISF 24218. That is, an SOP of this generic class may be parsed from INSTB 20262 and a corresponding SDP provided from FUDISF 24218 during a single system clock cycle. Operation of FUDISF 24218 thereby enhances speed of operation of JP 10114, in particular at the beginning of execution of new SOPs.

The second class of microinstruction operations are those specific to particular SINTs or to particular groups of SINTs. These groups of SINTs may reside entirely within a particular dialect, for example FORTRAN, or may exist within one or more dialects. SDPs for this class of microinstruction operation are provided by AF 24220. As described further below, AF 24220 is slower than FUDISF 24218, but is larger. In general, AF 24220 contains SDPs of microinstruction sequences specific to particular SINTs. In general, generic microinstruction operations are performed before those operations specific to particular SINTs, so that SDPs are required from AF 24220 at a later time than those from FUDISF 24218. SDPs for specific SINT operations may therefore be provided from lower speed AF 24220 without a penalty in speed of execution of SOPs.

Referring again to FUDISF 24218, FUDISF 24218 is a 1,024 word by 6 bit fast access by polar memory. Each word contained therein, as described above, is an SDP, or address to start of a corresponding sequence of microinstructions in FUSITT 11012. As will be described further below, FUSITT is an 8K (8192) word memory. SDPs provided by FUDISF 24218 are each, as described above, 6 bits wide and may thus address a limited, 32 word area of FUSITT 11012's address space. FUDISF 24218 is enabled to provide SDPs to FUSITT 11012 by microinstruction control signals (not shown for clarity of presentation) from FUSITT 11012. FUDISF 24218 six bit SDPs are encoded by FUDISENC 24219 to address FUSITT 11012 address space in increments of 4 microinstructions, that is in increments of 4 address locations. FUDISF 24218 SDPs thereby address 4 microinstructions at a time from FUSITT 11012's microinstruction sequences. As will be described further below, mPC 20276 generates successive microinstruction addresses to FUSITT 11012 to select successive microinstructions of a sequence following an initial microinstruction selected by an SDP from FUSDT 11010. An FUDISF 24218 SDP will thereby select the first microinstruction of a 4 microinstruction block, and mPC 20276 will select the following 3 microinstructions of that 4 microinstruction sequence. A 4 microinstruction sequence may therefore be executed in line, or sequentially, for each FUDISF 24218 SDP provided in response to a generic SOP. FUDISENC 24219 encodes FUDISF 24218 six bit SDPs to select these 4 microinstruction sequences so that the least significant bit of these SDPs occupies the 24 bit of FUSITT 11012 address inputs, and so on. The two least significant bits of an FUSITT 11012 address, or SDP, provided from FUDISF 24218 are forced to 0 while the ninth and higher bits may be hard-wired to define any particular block of 128 addresses in FUSITT 11012. This hard-wiring of the most significant bits of FUSITT 11012 addresses from FUDISF 24218 allows a set of generic microinstruction sequences selected by FUDISF 24218 to be located as desired within FUSITT 11012's address space. FUDISENC 24219 is comprised of a set of driver gates.

As previously described, SDPs for generic microinstructions currently being utilized by CS 10110 in executing user's programs are written into FUDISF 24218 from Bits 10 to 15 of JPD Bus 10142 (JPD(10-15)). Addresses for loading SDPs into FUDISF 24218 are provided, as previously described, from LADDR 24214. LADDR 24214 is enabled to provide load addresses, and FUDISF 24218 is enabled to be

written into, by microinstruction control signals (not shown for clarity of presentation) provided from FUSITT 11012.

Referring to AF 24220, as previously described AF 24220 is of larger capacity than FUDISF 24218, but has slower access time. AF 24220 is a 1,024 word by 15 bit memory. In general, 2 clock cycles are required to obtain a DSP from AF 24220. During first clock cycle, an SOP is loaded into LOPCODE 24210 and, during second clock cycle, AF 24220 is addressed to provide a corresponding SDP. SDPs provided by AF 24220 are each 15 bits in width and thus capable of addressing a larger address space than that of FUSITT 11012. As previously described, FUSITT 11012 is an 8K word memory. If FUSITT 11012 is addressed by an AF 24220 SDP referring to an address location outside of FUSITT 11012's address space, FUSITT 11012 will generate a microinstruction Not In Control Store output to EVENT 20284 as described further below. An AF 24220 SDP resulting in this event will then be used to address certain microinstruction sequences stored in MEM 10112. These microinstructions will then be executed from MEM 10112, rather than from FUSDT 11010. This operation allows certain microinstruction sequences, for example rarely used microinstruction sequences, to remain in MEM 10112, thus freeing AF 24220 and FUSITT 11012's address spaces from more frequently used SOPs.

As previously described AF 24220 is loaded, with SDPs, for SINTs currently being used by CS 10110 in executing user's programs, from Bits 16—31 of JPD Bus 10142 (JPD(16—31)). Also as previously discussed, addresses to load SDPs into AF 24220 are provided from LADDR 24214. LADDR 24214 is enabled to provide load addresses and AF 24220 to receive SDPs, by microinstruction control signals (not shown for clarity of presentation) provided from FUSITT 11012.

Referring finally to EUSDT 20266, SDPs may be provided to EU 10122 from 3 sources. EU 10122 SDPs may be provided from EUDISF 24222, from JPD Bus 10142 or from literal fields of microinstructions provided from FUSITT 11012. EUDISF 24222's SDPs are each 12 bits in width and comprise 9 bits of address into EUSITT and 3 bits of operand format information.

EUDISF 24222 is 1,024 word by 12 bit memory. As previously described addresses to read SDPs from EUDISF 24222 are provided from OPCODEREG 20268 by concatenating a 4 bit Dialect Code from RDIAL 24212 and an 8 bit SOP from LOPCODE 24210. SDPs provided by EUDISF 24222 are provided as a first input to EUDISS 24224.

EUDISS 24224 is a multiplexer. As just described, a first input of EUDISS 24224 are SDPs from EUDISF 24222. A second 12 bit input of EUDISS 24224 is provided from Bits 20 to 31 of JPD Bus 10142 (JPD(20—31)). A third input of EUDISS 24224 is a 12 bit input provided from a literal field of an FUSITT 11012 microinstruction output. EUDISS 20224 selects one of these 3 inputs to be transferred on EUDIS Bus 20206 to be provided as an execute unit SDP to EUSITT. Selection between EUDISS 20224's inputs is provided by microinstruction control signals (not shown for clarity of presentation) provided from FUSITT 11012.

As previously described, EUDISF 24222 is loaded, with SDPs for S-Language dialects currently being used by CS 10110, from Bits 20 to 31 of JPD Bus 10142 (JPD(20—31)). Addresses to load SDPs into EUDISF 24222 are provided, as previously described, from LADDR 20214. FUSITT 11012 provides enable signals (not shown for clarity of presentation) to LADDR 24214 and EUDISF 24222 to enable writing of SDPs into EUDISF 24222.

The structure and operation of FUCTL 20214 circuitry for fetching and parsing SINTs from MEM 10112 to provide Name syllables and SOPs, and for interpreting SOP to provide SDPs to FUSITT 11012 and EUSITT from FUSDT 11010 and EUSDT 20266, have been described above. As described above, SDPs provided by FUSDT 11010 and EUSDT 20266 are initial, or starting, addresses pointing to first microinstructions of sequences of microinstructions. Addresses for microinstructions following those initial microinstructions are provided by FUCTL 20214's next address generator circuitry which may include mPC 20276, BRCASE 20278, REPCTR 20280 and PNREG 20282, EVENT 20284 and SITTNAG 20286. mPC 20276, BRCASE 20278, REPCTR 20280 and PNREG 20282, and SITTNAG 20286 are primarily concerned with generation of next addresses during execution of microinstruction sequences in response to SOPs and will be described next below. EVENT 20284 and other portions of FUCTL 20214's circuitry are more concerned with generation of microinstruction sequences with regard to CS 10110's internal mechanisms operations and will be described in a later description. EU 10122 also includes next address generation circuitry and this circuitry will be described in a following description of EU 10122.

c.c.c. Next Address Generator 24310 (Fig. 243)

As stated above, in EU 10120 first, or initial, microinstructions of microinstruction sequences for interpreting SOPs are provided by FUSDT 11010. Subsequent addresses of microinstructions within these sequences are, in general, provided by mPC 20276 and BRCASE 20278. mPC 20276, as described further below, provides sequential addresses for selecting sequential microinstructions of microinstruction sequences. BRCASE 20278 provides addresses for selecting microinstructions when a microinstruction Branch or microinstruction Case operation is required. REPCTR 20280 and PNREG 20282 provide addresses for writing, or loading, of microinstruction sequences into FUSITT 11012. Other portions of FUCTL 20214 circuitry, for example EVENT 20284, provides microinstruction sequence selection addresses to select microinstruction sequences for controlling operation of CS 10110's internal mechanisms. SITTNAS 20286 selects between these microinstruction address sources to provide to FUSITT 11012 those addresses

required to select microinstructions of the operation to be currently executed by CS 10110.

Referring to Fig. 243, a partial block diagram of FU 10120's Next Address Generator (NAG) 24310 is shown. In addition to FUSDT 11010, NAG 24310 includes mPC 20276, BRCASE 20278, EVENT 20284, REPCTR 20280 and PNREG 20282, a part of RCWS 10358, and SITTNAS 20286. EVENT 20284 is, as described above, primarily concerned with execution of microinstruction sequences for controlling CS 10110 internal mechanisms. EVENT 20284 as shown herein only to illustrate its relationships to other portions of NAG 24310. EVENT 20284 will be described further in a following description of FUCTL 20214's circuitry controlling CS 10110's internal mechanisms. Similarly, operation of RCWS 10358 will be described in part in the present description of NAG 24310, and in part in a following description of control of CS 10110's internal mechanisms.

Referring first to NAG 24310's structure, interconnections of FUSDT 11010, RCWS 10358, mPC 20276, BRCASE 20278, REPCTR 20280, PNREG 20282, EVENT 20284, and SITTNAS 20286 have been previously described with reference to Fig. 202. NAG 24310's structure will be described below only wherein Fig. 243 differs from Fig. 202.

Referring first to SITTNAS 20286, SITTNAS 20286 is shown as comprised of EVENT Gate (EVNTGT) 24310 and Next Address Select Multiplexer (NASMUX) 24312. NASMUX 24312 is comprised of NAS Multiplexer A (NASMUXA) 24314, NASMUXB 24316, NASMUXC 24318, and NASMUXD 24320. Outputs of EVNTGT 24310 and NASMUXA 24314 to NASMUXD 24320 are ORed to CSADR 20204 to provide microinstruction selection addresses to FUSITT 11012.

ADR 20202 is shown in Fig. 243 as comprised of nine buses, Address A (ADRA) Bus 24322 to Address I (ADRI) Bus 24338. Output of EVENT 20284 is connected to input of EVNTGT 24310 by ADRA Bus 24322. Outputs of REPCTR 20280 and PNREG 20282 and output of AF 24220 are connected to inputs of NASMUXA 24314 by, respectively, ADRB Bus 24324 and ADRC Bus 24326. Outputs of RCWS 10358 and FUDISENC 24219 are connected to inputs of NASMUXB 24316 by, respectively, ADRD Bus 24328 and ADRE Bus 24330. Outputs of BRCASE 20278 and second output of mPC 20276 are connected to inputs of NASMUXC 24318 by, respectively, ADRF Bus 24332 and ADRG Bus 24334. Second output of mPC 20276 and JAM output of NC 10226 are connected to inputs of NASMUXD 24320 by, respectively, ADRH Bus 24336 and ADRI Bus 24338. ADR 20202 thus comprises a set of buses connecting microinstruction address sources to inputs of SITTNAS 20286.

Referring to mPC 20276, mPC 20276 is comprised of Micro-Program Counter Counter (mPCC) 24340 and Micro-Program Counter Arithmetic and Logic Unit (mPCALU) 24342. Data input of mPCC 24340 is connected from CSADR Bus 20204. Output of mPCC 24340 is connected to a first input of mPCALU 24342 and is mPC 20276's third output to BRCASE 20278. Second input of mPCALU 24342 is a fifteen binary number set, for example by hard-wiring, to be binary one. Output of mPCALU 24342 comprises mPC 20276's first output, to RCWS 10358, and mPC 20276's second output, to inputs of NASMUXC 24318 and NASMUXD 24320.

BRCASE 20278 is shown in Fig. 243 as comprising Mask and Shift Multiplexer (MSMUX) 24344, Case Mask and Shift Logic (CASEMS) 24346, Branch and Case Multiplexer (BCMUX) 24348 and Branch and Case Arithmetic and Logic Unit (BCALU) 24350. A first input of MSMUX 24344 (AONBC, not previously shown) is connected from output of AONGRF 20232. A second input of MSMUX 24344 (OFFMUXR, not previously shown) is connected from output of OFFMUXR 23812. Output of MSMUX 24344 is connected to input CASEMS 24346, and output of CASEMS 24346 is connected to a first input of BCMUX 24348. A second input of BCMUX 24348, BLIT is connected from a literal field output of FUSITT 11012's microinstruction output. Output of BCMUX 24348 and third output of mPC 20276, from output of mPCC 24340, are connected, respectively, to first and second inputs of BCALU 24350. Output of BCALU 24350 comprises BRCASE 20278 outputs to NASMUXC 24318.

An address to select a next microinstruction may be provided to FUSITT 11012 by SITTNAS 20286 from any of eight sources. First source is output of mPC 20276. Output of mPC 20276 is referred to as Micro-Program Count Plus 1 (mPC+1) and is fifteen bits of address. Second source is from EVENT 20284 and is comprised of five bits of address. Third source is output of FUDISP 24218 and FUDISENC 24219 and, as previously described, is comprised of six bits of address. Fourth source is output of AF 24220 and, as previously described, is comprised of fifteen bits of address. Fifth source is output of BRCASE 20278. Output of BRCASE 20278 is referred to as Branch and Case Address (BRCASEADR) and comprises fifteen bits of address. Sixth source is an output of RCWS 10358. Output of RCWS 10358 is referred to as RCWS Address (RCWSADR) and is comprised of fifteen bits of address. Seventh source is REPCTR 20280 and PNREG 20282 whose outputs (REPPN) together comprise fifteen bits of address. Finally, eighth source is JAM input from NC 10226, which comprises five bits of address. These address sources differ in number of bits of address that they provide, but a microinstruction address gated onto CSADR Bus 20204 by SITTNAS 20286 always comprises fifteen bits of address. If a particular source applies fewer than fifteen bits, that address is extended to fifteen bits by SITTNAS 20286. In general, extension of address bits may be performed by hard-wiring of additional address input bits to SITTNAS 20286 from each of these sources and will be described further below.

Referring to mPC 20276, mPCC 24340 is a fifteen bit register and mPCALU 24342 is a fifteen bit ALU. mPCC 24340 is, as described above, connected from CSADR Bus 20204 and is sequentially loaded with a microinstruction address currently being presented to FUSITT 11012. mPCC 24340 will thus contain the

address of the currently executing microinstruction. mPCALU 24342 is dedicated to incrementing the address contained in mPCC 24340 by one. mPC+1 output of mPCALU 24342 will thereby always be address of next sequential microinstruction. mPC+1 is, as described above, a fifteen bit address and is thus not extended in SITNAS 20286.

6 Referring to BRCASE 20278, as described above BRCASE 20278 provides next microinstruction addresses for mPC 20276 Relative Branches and for Case Branches. Next microinstruction addresses for microprogram Relative Branches and for Case Branches are both generated as addresses relative to address of currently executing microinstruction as stored in mPCC 24340, but differ in the manner in which these relative addresses are generated. Considering first Case Branches, Case Branch addresses relative to a currently executing microinstruction address are generated, in part, by MSMUX 24344 and CASEMS 24346. As described above, MSMUX 24344 which is a multiplexer receives two inputs. First input is AONBC from output of AONGRF 20232 and second input is OFFMUXR from output of OFFMUXR 23812. Each of these inputs is eight bits, or one byte, in width. Acting under control of microinstruction output from FUSITT 11012, MSMUX 24344 selects either input AONBC or input OFFMUXR as an eight bit output to input 10 of CASEMS 24346. CASEMS 24346 is a Mask and Shift circuit, similar in structure and operation to that of FIU 20116 but operating upon bytes rather than thirty-two bit words. CASEMS 24346, operating under microinstruction control from FUSITT 11012, manipulates eight bit input from MSMUX 24344 by masking and shifting to provide eight bit Case Value (CASEVAL) output to BCMUX 24348. CASEVAL represents a microinstruction address displacement relative to address of a currently executing microinstruction and, 15 being an eight bit number, may express a displacement of 0 to 255 address locations in FUSITT 11012.

20 BCMUX 24348 is an eight bit multiplexer, similar in structure and operation to MSMUX 24344, and is controlled by microinstruction inputs provided from FUSITT 11012. In executing a case operation, BCMUX 24348 selects CASEVAL input to MCMUX 24348's output to first input of BCALU 24350. BCALU 24350 is a sixteen bit arithmetic and logic unit. Second input of BCALU 24350 is fifteen bit address of currently 25 executing microinstruction from mPCC 24340. BCALU 24350 operates under microinstruction control provided from FUSITT 11012 and, in executing a Case operation, adds CASEVAL to the address of a currently executing microinstruction. During a Case operation, carry input of BSALU 24350 is forced, by microinstruction control from FUSITT 11012, to one so that BCALU 24350's second input is effectively 30 mPC+1, or address of currently executing microinstruction plus 1. Output BRCASEADR of BCALU 24350 will thereby be fifteen bit Case address which is between one and 256 FUSITT 11012 address locations higher than the address location of the currently executing microinstruction. The actual case value address displacement from the address of the currently executing microinstruction is determined by either input AONBC or input OFFMUXR to MSMUX 24344, and these mask and shift operations are performed by CASEMS 24346.

35 Case operations as described above may be used, for example, in interpreting and manipulating CS 10110 table entries. For example, Name Table Entries of Name Tables 10350 contain flag fields carrying information regarding certain operations to be performed in resolving and evaluating those Name Table Entries. These operations may be implemented as Case Branches in microinstruction sequences for resolving and evaluating those Name Table Entries. In the present example, during resolve of a Name 40 Table Entry the microinstruction sequence for performing that resolve may direct a byte of that Name Table Entry's flag field to be read from AONGRF 20232, or OFFMUXR 23812, and through MSMUX 24344 to CASEMS 24346. That microinstruction sequence will then direct CASEMS 24346 to shift and mask that flag field byte to provide a CASEVAL. That CASEVAL will have a value dependent upon the flags within that flag field byte and, when added to mPC+1, will provide a FUSITT 11012 microinstruction address for a 45 microinstruction sequence for handling that Name Table Entry in accordance with those flag bits.

As described above, BRCASE 20278 may also generate microinstruction addresses for Branches occurring within execution of a given microinstruction sequence. In this case, microinstruction control signals from FUSITT 11012 direct BCMUX 24348 to select BCMUX 24348's second input as output to BCALU 24350. BCMUX 24348's second input is Branch Literal (BLIT). As described above, BLIT is provided from a 50 literal field of a microinstruction word from FUSITT 11012's microinstruction output. BLIT output of BCMUX 24348 is added to address of currently executing microinstruction from mPCC 24340, and BCALU 24350, to provide fifteen bit BRCASEADR of a microinstruction address branched to from the address of the currently executing microinstruction. BRCASEADR may represent, for example, any of four Branch Operations. Possible Branch Operations are: first, a Conditional Short Branch; second, a Conditional Short Call; third, a 55 Long Go To; and, fourth, a Long Call. In each of these possible Branch Operations, BLIT is treated as the two's complement of the desired branch value, that is the microinstruction address offset relative to the address of the currently executing microinstruction. BLIT field may therefore be, effectively, added to or subtracted from the address of the currently executing microinstruction, to provide a microinstruction address having a positive or negative displacement from the address of the currently executing 60 microinstruction. In a Conditional Short Branch or a Conditional Short Call, the fourteen bit literal field is a sign extended eight bit number. Both Conditional Short Branch and Conditional Short Call microinstruction addresses may therefore point to an address within a range of +127 to -128 FUSITT 11012 address locations of the address of the currently executing microinstruction. In the case of a Long Go To or Long Call, the BLIT field is a fourteen bit number representing displacement relative to the address of the 65 currently executing microinstruction. BRCASEADR may, in these cases, represent a FUSITT 11012

microinstruction address within a range of +8191 to -8192 FUSITT 11012 address locations of the address of the currently executing microinstruction. BRCASE 20278 thereby provides FU 10120 with capability of executing a full range of microinstruction sequence Case and Branch operations.

Referring to RCWS 10358, as previously described RCWS 10358 stores information regarding microinstruction sequences whose execution has been halted. RCWS 10358 allows execution of those microinstruction sequences to be resumed at a later time. A return control word (RCW) may be written onto RCWS 10358 during any microinstruction sequence that issues a Call to another microinstruction sequence. The calling microinstruction sequence may, for example, be aborted to service an event, as described further in a following description, or may result in a Jam. A Jam is a call for a microinstruction sequence which is forced by operation of CS 10110 hardware, rather than by a microinstruction sequence. RCWS 10358 operation with regard to CS 10110's internal mechanisms will be described in a following description of EVENT 20284, STATE 20294, and MCW1 20290 and MCWO 20292. For purposes of the present discussion, that portion of a RCW concerned with interpretation of SOPs contains, first, certain state information from FUSITT 11012 and, second, a return address into FUSITT 11012. State that FUSITT 11012 state is provided from STATE 20294, as described below, and that portion of a RCW containing FUSITT 11012 state information will be described in a following description. Microinstruction address portions of RCWs are provided from output of mPCALU 24342. This microinstruction address is the address of the microinstruction to which FU 10120 is to return upon return from a Call, Event, or Jam. Upon occurrence of a Call or Jam, the microinstruction return address is mPC+1, that is the address of the microinstruction after the microinstruction issuing the Call or Return. For aborted microinstruction sequences, the microinstruction return address is mPC, that is the address of the microinstruction executing at the time abort occurs.

Upon return from a call, service of an event, or service of a jam, FU 10120 state flag portion of RCW is loaded into STATE 20294. Microinstruction return address is provided by RCWS 10358 as fifteen bit RCWSADR to SITNAS 20286 and is gated onto CSADR 20204. RCWSADR is provided to FUSITT 11012 to select the next microinstruction and is loaded into mPCC 24340 from CSADR 20204.

As previously described, RCWS 10358 is connected to JPD Bus 10142 by a bi-directional bus. RCWs may be written into RCWS 10358 from JPD Bus 10142, or read from RCWS 10358 to JPD Bus 10142. The fifteen bit next microinstruction address portion, and the single bit FUSITT 11012 state portion of RCW is written from or read to Bits 16 to 31 of JPD Bus 10142. FU 10120 may write Present Bottom RCW or Previous RCW into RCWS 10358 from JPD Bus 10142 and may read Present Bottom RCW, or Previous RCW, or another selected RCW, onto JPD Bus 10142. RCWS 10358 thereby provides a means for storing and returning microinstruction addresses of microinstruction sequences whose execution has been suspended, and a means for writing and reading microinstruction address, and FUSITT 11012 state flags, from and to JPD Bus 10142.

As previously described, REPCTR 20280 and PNREG 20282 provide microinstruction addresses for writing of microinstructions into FUSITT 11012. REPCTR 20280 is an eight bit counter and PNREG 20282 is a seven bit register. Eight bit output of REPCTR 20280 is left concatenated with seven bit output of PNREG 20282 to provide fifteen bit microinstruction addresses REPPN. That is, REPCTR 20280 provides the eight low order bits of microinstruction address while PNREG 20282 provides the seven most significant bits of address.

REPCTR may be loaded from Bits 24—31 of JPD Bus 10142, and may be read to Bits 24—31 of JPD Bus 10142. In addition, the eight bits of microinstruction address in REPCTR 20280 may be incremented or decremented as microinstructions are written into FUSITT 11012.

As described above, PNREG 20282 contains the seven most significant bits of microinstruction address. These address bits may be written into PNREG 20282 from Bits 17—23 of JPD Bus 10142. Contents of PNREG 20282 may not, in general, be read to JPD Bus 10142 and may not be incremented or decremented.

Referring to JAM input to SITNAS 20286 from NC 10226, certain Name evaluate or resolve operations may result in jams. A Jam functions as a call to microinstruction sequences for servicing Jams and are forced by FU 10120 hardware circuitry involved in Name syllable evaluates and resolves.

JAM input to SITNAS 20286 is comprised of six Jam address bits. Three bits are provided by NC 10226 and three bits are provided from FUSITT 11012's microinstruction output as part of microinstruction sequences for correcting Name syllable evaluates and resolves. The three bits of address from NC 10226 form the most significant three bits of JAM address. One of these bits gates JAM address onto CSADR Bus 20204 and is thus not a true address bit. Output of FUSITT 11012 provides the three least significant bits of JAM address and specifies the particular microinstruction sequence required to service the particular Jam which has occurred. Therefore, during Name evaluate or resolves, the microinstruction sequences provided by FUSITT 11012 to perform Name evaluates or resolves specifies what microinstruction sequences are to be initiated if a Jam occurs. The three bits of JAM address provided by NC 10226 determine, first, that a Jam has occurred and, second, provide two bits of address which, in combination with the three bits of address from FUSITT 11012, specify the particular microinstruction sequence for handling that Jam. JAM address inputs from NC 10226 and from FUSITT 11012 thereby provide six of the fifteen bits of JAM address. The remaining nine bits of JAM address are provided, for example, by hard-wired inputs to NASMUXD 24320. These hard-wired address bits force JAM address to address FUSITT

11012 in blocks of 4 microinstruction addresses, in a manner similar to address inputs to FUDISF 24218 and FUDISENC 24219.

Address inputs provided to SITTNAS 20286 from FUSDT 11010 have been previously described with respect to description of FUCTL 20214 fetch, parse, and dispatch operations. Address inputs provided by
 5 EVENT 20284 will be described in a following description of FUCTL 20214's operations with regard to CS 10110's internal mechanisms.

Referring finally to SITTNAS 20286, as previously described SITTNAS 20286 is comprised of EVNTGT 24310 and NASMUX 24312. Inputs are provided to NASMUX 24312, as described above, from FUSDT 11010, mPC 20276, BRCASE 20278, RCWS 10358, REPCTR 20280 and PNREG 20282, and by JAM input.
 10 These inputs are, in general, provided with regard to FUCTL 20214's operations in fetching, parsing, and interpreting SOPs and Name syllables. These operations are thereby primarily directly concerned with execution of user's programs, that is the execution of sequences of SINS. NASMUX 24312 selects between these inputs and transfers selected address inputs onto CSADR 20204 as microinstruction addresses to FUSITT 11012 under microinstruction control from microinstruction outputs of FUSITT 11012.
 15 Microinstruction address outputs are provided to SITTNAS 20286 from EVENT 20284 in response to Events, described further below, occurring in CS 10110's operations in executing user's programs. These microinstruction addresses from EVENT 20284 are gated onto CSADR 20204, to select appropriate microinstruction sequences, by EVNTGT 24310. EVNTGT 24310 is separated from NASMUX 24312 to allow EVNTGT 24310 to over-ride NASMUX 24312 and provide microinstruction address to EVENT 20284 while
 20 NASMUX 24312 is inhibited due to occurrence of certain Events. These Events are, in general, associated with operation of CS 10110's internal mechanisms and structure and operation of EVENT 20284, together with STATE 20294, MCW1 20290, and MCWO 20292, and other portions of RCWS 10358, will be described next below.

25 c.c. FUCTL 20214 Control Circuitry for CS 10110 Internal Mechanisms (Figs. 244—249)

Certain portions of FUCTL 20214's Control Circuitry are more directly concerned with operation of CS 10110's internal mechanisms, for example CS 10110 Stack Mechanisms. This circuitry may include STATE 20294, EVENT 20284, MCW1 20290 and MCWO 20292, portions of RCWS 10358, REG 20288, and Timers 20296. These FUCTL 20214 control elements will be described next below, beginning with STATE 20294.

30 a.a.a. State Logic 20294 (Figs. 244A—244Z)

In general, all CS 10110 operations, including execution of microinstructions, are controlled by CS 10110's Operating State. CS 10110 has a number of Operating States, hereafter referred to as States, each State being defined by certain operations which may be performed in that State. Each of these States will
 35 be described further below. Current State of CS 10110 is indicated by a set of State Flags stored in a set of registers in STATE 20294. Each State is entered from previous State and is exited to a following State. Next State of CS 10110 is detected by random logic gating distributed throughout CS 10110 to detect certain conditions indicating which State CS 10110 will enter next. Outputs of these Next State Detection gates are provided as inputs to STATE 20294's registers. A particular State register is set and provides a State Flag
 40 output when CS 10110 enters the State associated with that particular register. State Flag outputs of STATE 20294's state registers are provided as enable signals throughout CS 10110 to enable initiation of operations allowed within CS 10110's current State, and to inhibit initiation of operations which are not allowed within CS 10110's current State.

Certain of CS 10110's States, and associated STATE 20294 State Registers and State Flag outputs, are:

45 (1) MO: the initial State of any microinstruction.

State MO is always entered as first data cycle of every microinstruction. During MO, CS 10110's State may not be changed, thus allowing a microinstruction to be arbitrarily aborted and restarted from State MO. In normal execution of microinstructions, State MO is followed by State M1, described below, that is, State MO is exited to State M1. State MO may be entered from State M0 and from State M1, State AB, State
 50 LR, State NR, or State MS, each of which will be described below.

(2) EP: Enable Pause State. State EP is entered when State MO is entered for the first time in a microinstruction. If that microinstruction requests a pause, that microinstruction will force State MO to be re-entered for one clock cycle. If State M0 lasts more than one clock cycle, State EP is entered on each extension of State M0 unless the extension is a result of a pause request.

55 (3) SR: Source GRF State. SR State is active for one clock cycle wherein SR State register enables loading of a GRF 10354 output register. State SR is re-entered on every State M0 cycle except a State M0 cycle generated by a microinstruction requesting extension of State M0. When all STATE 20294 State Registers are cleared, DP 20218 may set state SR register alone, for purposes of reading from GRF 10354.

(4) M1: Final state of normal microinstruction execution. State M1 is the exit State of normal microinstruction execution. FUSITT 11012 microinstruction register, described below, is loaded with a next microinstruction upon exit from State M1. In addition, State M1 Flag output of STATE 20294 enables all CS
 60 10110 registers to receive data on their inputs, that is data on inputs of these registers are clocked to outputs of these registers. State M1 may be entered from State M1, or from State M0, State MW, State MWA, or State WB.

65 (5) LA: Load Accumulator Enable State. State LA is entered, upon exit from State M1, by

EP 0 067 556 B1

microinstructions which read data from MEM 10112 to OFFMUXR 23812. As previously described, OFFMUXR 23812 serves as a general purpose accumulator for DESP 20210. STATE LA overlaps into execution of next microinstruction, and persists until data is returned from MEM 10112 in response to a request to MEM 10112. When MEM 10112 signals data is available, by asserting DAVFA, LA State Flag enables loading of data into OFFMUXR 23812. If the next microinstruction references OFFMUXR 23812, that microinstruction execution is deferred until a read to OFFMUXR 23812 is completed, as indicated by CS 10110 exiting from State LA.

(6) RW: Load GRF 10354 Wait State. State RW is entered from State M1 of microinstructions which read data from MEM 10112 to GRF 10354. RW Flag inhibits initiation of a next microinstruction, that is prevents entry to State M0, and persists through the CS 10110 clock cycle during which data is returned from MEM 10112 in response to a request. State RW initiates Load GRF Enable State, described below.

(7) LR: Load GRF Enable State. State LR is entered in parallel with State RW, on last clock cycle of RW, and persists for one CS 10110 clock cycle. LR Flag enables writing of MEM 10112 output data into GRF 10354.

(8) MR: Memory Reference Trailer State. State MR is entered on transition to State M0 whenever a previous microinstruction makes a logical or physical address reference to MEM 10112. MR Flag enables recognition of any MEM 10112 reference Events, described below, which may occur. State MR persists for one clock cycle. If an MEM 10112 memory reference Event occurs, that Event forces exit from State MR to States AB and MA, otherwise State MR has no effect upon selection next state.

(9) SB: Store Back Enable State. State SB is entered during State M0 of a microinstruction following a microinstruction which generated a store back of a result of a EU 10122 operation. SB Flag gates that result to be written into MEM 10112 through JPD Bus 10142.

(10) AB: Microinstruction Abort State. State AB is entered from first M0 State after an Event request is recognized, as described in a following description.

State AB may be entered from State M0 or from State AB and suppresses an entry into State M1. If there has been an uncompleted reference to MEM 10112, that is, the reference has not been aborted and data has not returned from MEM 10112, JP 10114 remains in State AB until the MEM 10112 reference is completed. Should an abort have occurred due to a MEM 10112 reference Event, State AB lasts two clock cycles only. As will be described in a following description of EVENT 20284, State M0 of a first microinstruction of a Handler for an Event causing an abort is entered from State AB. AB Flag gates the Handler address of the highest priority recognized Event onto CSADR Bus 20204 to select a corresponding Event Handler microinstruction sequence. EVENT 20284 is granted control of CSADR Bus 20204 during all State AB clock cycles.

(11) AR: Microinstruction Abort Reset State. State AR is entered in parallel with first clock cycle of State AB and persists for one clock cycle. AR Flag resets various STATE 20294 State Registers when an abort occurs. If there are no uncompleted MEM 10112 references, next State AB clock cycle is the last. On uncompleted MEM 10112 references, State AR is entered, but State AB remains active until reference is complete. Should a higher priority Event request service and be recognized while JP 10114 is in State AB, State AR is reentered. State AB will thereby be active for two clock cycles during all honored Event requests.

(12) MA: MEM 10112 Reference Abort. State MA is entered in parallel with State AB if a MEM 10112 reference is aborted, as indicated by asserted ABORT control signal output from MEM 10112. State MA persists for one clock cycle and State AB flag generates a MEM 10112 Reference Abort Flag which, as described below, results in a repeat of the MEM 10112 reference. AB Flag also resets MEM 10112 Trailer States, described below.

(13) NW: Nano-interrupt Wait State. State NW is entered from State M0 of a microinstruction which issues a Nano-interrupt Request to EU 10122 for an EU 10122 operation. FU 10120 remains in State NW until EU 10122 acknowledges that interrupt. Various EU 10122 Events may make requests at this time. State NW is exited into State AB or State M1.

(14) FM: First Microinstruction of a SIN. State FM is entered in parallel with State M0 on first microinstruction of each SIN and persists for one clock cycle. FM Flag inhibits premature use of AF 24220 and enables recognition of SIN Entry Events. State FM is re-entered upon return from all aborts taken during State M0 of the first microinstruction of an SIN.

(15) SOP: Original Entry to First SIN. State SOP is entered upon entry to State M0 of the first microinstruction of an SOP and is exited from upon any exit from that microinstruction. State SOP is entered only once for each SOP. SOP Flag may be used, for example, for monitoring performance of JP 10114.

(16) EU: EU 10122 Operand Buffer Unavailable. State EU is entered from State M0 of a microinstruction which attempts to read data to EU 10122 Operand Buffer, described in a following description, wherein EU 10122 Operand Buffer is full. When a new SOP is entered, three fetches of data from MEM 10112 may be performed before EU 10122 Operand Buffer is full; two fetches will fill EU 10122 Operand Buffer but EU 10122 may take one operand during a second fetch, thereby clearing EU 10122 Operand Buffer space for a third operand.

(17) NR: Long Pipeline Read. Entry into State NR disables overlap of MEM 10112 reads and disables execution of the next microinstruction. A following microinstruction does not enter State M0 until ...

requested data is returned from MEM 10112. State NR is entered from State NR or from State M1.

(18) NS: Nonpipeline Store Back. State NS is entered in parallel with State SB whenever a microinstruction requesting a pipeline store back, or a write to MEM 10112, occurs. State NS flag generates entry into State M0 of a following microinstruction upon exit from State SB.

5 (19) WA: Load Control Store State A. State WA is entered from State M0 of a microinstruction which directs loading of microinstruction into FUSITT 11012. WA State Flag controls selection of addresses to CSADR Bus 20204 for writing into FUSITT 11012, and generates a write enable pulse to FUSITT 11012 to write microinstructions into FUSITT 11012.

10 (20) WB: Load Control Store State B. State WB is entered from State WA and is used to generate an appropriate timing interval for writing into FUSITT 11012. State WB also extends State M1 to 2 clock cycles to ensure a valid address input to FUSITT 11012 when a next microinstruction is to be read from FUSITT 11012.

15 Having described certain CS 10110 states, and operations which may be performed within those states, state sequences for certain CS 10110 operations will be described next below with aid of Figs. 244A to 244Z. Fig. 244A to Fig. 244Z represent those state timing sequences necessary to indicate major features of CS 10110 state timing. All state timing shown in Figs. 244A to 244V assumes full pipelining of CS 10110 operations, for example pipelining of reads from and writes to MEM 10112 by JP 10114. Pipelining is not assumed in Figs. 244W to 244Z. Referring to Figs. 244A to 244Z, these figures are drawn in the form of timing diagrams, with time increasing from left to right. Successive horizontally positioned "boxes" represents successive CS 10110 states during successive CS 10110 110 nano-second clock cycles. Vertically aligned "boxes" represent alternate CS 10110 states which may occur during a particular clock cycle. Horizontally extended dotted lines connecting certain states represented in Fig. 244A to 244Z represent an indeterminate time interval which is an integral multiple of 110 nano-second CS 10110 clock cycles.

Referring to Fig. 244A to 244Z in sequence, State Timing Sequences shown therein represent:

25 (1) Fig. 244A; state timing for execution of a normal microinstruction with no Events occurring and no MEM 10112 references.

(2) Fig. 244B execution of a normal microinstruction, with no Events occurring, no MEM 10112 references, and a hold in State M0 for one clock cycle.

30 (3) Fig. 244C; a microinstruction requests an extension of State M0 for one clock cycle, with no Events occurring and no MEM 10112 references.

(4) Fig. 244D; a write to MEM 10112 from DESP 20210, for example from GRF 10354 or from OFFALU 20242. MEM 10112 port is available and MEM 10112 reference is made during first sequential occurrence of States M0 and M1.

35 (5) Fig. 244E; a write to MEM 10112 from DESP 20210 as described above. MEM 10112 port is unavailable for an indeterminate number of clock cycles. A MEM 10112 reference is made during first sequential occurrence of States M0 and M1.

(6) Fig. 244F; writing of an EU 10122 result back into MEM 10112. MEM 10112 is available and a write operation is initiated during first sequential occurrence of States M0 and M1.

40 (7) Fig. 244G; writing back of an EU 10122 result to MEM 10112 as described above. MEM 10112 port is unavailable for an undetermined number of clock cycles, or EU 10122 does not have a result ready to be written into MEM 10112. Write operation is initiated during first sequential occurrence of States M0 and M1.

(8) Fig. 244H; a read of an EU 10122 result into FU 10120. EU 10122 result is not available for an undetermined number of clock cycles.

45 (9) Fig. 244I; a read from MEM 10112 to OFFMUXR 23812, with no delays. The microinstruction following the microinstruction initiating a read from MEM 10112 does not reference OFFMUXR 23812.

(10) Fig. 244J; a read from MEM 10112 to OFFMUXR 23812 with data from MEM 10112 being delayed by an indeterminate number of clock cycles. The next following microinstruction from that initiating the read from MEM 10112 does not reference OFFMUXR 23812.

50 (11) Fig. 244K; a read from MEM 10112 to OFFMUXR 23812. The next microinstruction following the microinstruction initiating the read from MEM 10112 references OFFMUXR 23812.

(12) Fig. 244L; a read from MEM 10112 to GRF 10354. The read to GRF 10354 is initiated by the first sequentially occurring States M0 and M1.

(13) Fig. 244M; a read from MEM 10112 to GRF 10354 and to OFFMUXR 23812. In this case, read operations may not be overlapped.

55 (14) Fig. 244N; JP 10114 honors an Event request and initiates a corresponding Event Handler microinstruction sequence, no MEM 10112 references occur.

(15) Fig. 244O; JP 10114 honors an Event request as stated above. MEM 10112 references are made during the first sequential occurrence of States M0 and M1 and a MEM 10112 reference Event occurs. In case of an MEM 10112 reference event, State MA is entered from one clock cycle. This occurs only if a MEM 10112 reference is made and aborted.

60 (16) Fig. 244P; an Event occurs in a MEM 10112 reference made during the first sequential occurrence of States M0 and M1. The MEM 10112 reference does not result in a memory reference Event. CS 10110 remains in State AB until the MEM 10112 reference is completed by return of data from MEM 10112.

65 (17) Fig. 244Q; a read of data from MEM 10112 or JPD Bus 10114 to EU 10122 Operand Queue. EU 10122 Operand Queue is not full.

(18) Fig. 244R; a read of MEM 10112 or JPD Bus 10142 data to EU 10122 Operand Queue. EU 10122 Operand Queue is full when the microinstruction initiating the read is issued.

(19) Fig. 244S; a request for a "nano-interrupt" to EU 10122 by FU 10120 with no Events occurring.

5 (20) Fig. 244T; FU 10120 submits a "nano-interrupt" request to EU 10122 and an EU 10122 State Overflow, described further in a following description, occurs. No other Events are recognized, as described in a following description of EVENT 20284.

(21) Fig. 244U; FU 10120 submits a "nano-interrupt" request to EU 10122. Another Event is recognized during State M0 and an abort results. First abort state is entered for the non-EU 10122 event. All aborts recognized in State M0 are taken or acknowledged, before entrance into State M0. Therefore, on retry at 10 State M0 of the original microinstruction entered from State M0, next abort recognized is for EU 10122 Stack Overflow Event since EU 10122 Stack Overflow has higher priority.

(22) Fig. 244V; a load of a 27 bit microinstruction segment into FUSITT 11012.

In Figs. 244A to 244V, pipelining MEM 10112 reads and writes, and of JP 10114 operations, has been assumed. In Figs. 244W to 244Z, non-overlapping operation of JP 10114 is assumed.

15 (23) Fig. 244W; a read of data from MEM 10112 to OFFMUXR 23812.

(24) Fig. 244X; a read of data from MEM 10112 to EU 10122 Operand Queue.

(25) Fig. 244Y; a write of an EU 10122 result into MEM 10112.

(26) Fig. 244Z; a read of a 32 bit SIN word from MEM 10112 in response to a prefetch or conditional prefetch request.

20 Having described the general structure and operation of STATE 20294, and the operating states and operations of CS 10110, structure and operation of EVENT 20284 will be described next below.

b.b.b. Event Logic 20284 (Figs. 245, 246, 247, 248)

25 An Event is a request for a change in sequence of execution of microinstructions which is generated by CS 10110 circuitry, rather than by currently executing microinstructions. Occurrence of an Event will result in provision of a microinstruction sequence, referred to as an Event Handler, by FUSITT 11012 which modifies CS 10110's operations in accordance with the needs of that Event. Event request signals may be generated by CS 10110 circuitry internal to JP 10114, that is from FU 10120 or EU 10122 or CS 10110 circuitry external to JP 10114, for example from IOP 10116 or from MEM 10112. Event request signals are 30 provided as inputs to EVENT 20284. As will be described further below, EVENT 20284 masks Event Requests to determine which Events will be recognized during a particular CS 10110 Operating State, assigns priorities for servicing multiple Event Requests, and fabricates Handler addresses to FUSITT 11012 for microinstruction sequences for servicing requests. EVENT 20284 then provides those Handler microinstruction addresses to FUSITT 11012 through EVNTGT 24310, to initiate execution of selected Event 35 Handler microinstruction sequences.

Certain terms and expressions are used throughout the following description. The following paragraphs define these usages and provide examples illustrating these terms. An Event "makes a request" when a condition in CS 10110 hardware operation results in a Event Request signal being provided to EVENT 20284. As will be described further below, these Event Request signals are provided to 40 EVENT 20284 combinatorial logic which determines the validity of those "requests".

An Event Request "is recognized" if it is not masked, that is inhibited from being acted upon. Masking may be explicit, using masks generated by FUSITT 11012, or may be implicit, resulting from an improper CS 10110 State or invalid due to other considerations. That is, certain Events are recognized only during certain CS 10110 States even though those requests may be recognized during certain other states. Any 45 number of requests, for example up to 31, may be simultaneously recognized.

An Event Request is "honored" if it is the highest priority Event Request occurring. When a request is honored, a corresponding address, of a corresponding microinstruction sequence in FUSITT 11012, for its Handler microinstruction sequence is gated onto CSADR Bus 20204 by EVENT 20284. A request is honored when CS 10110 enters State AB. State AB gates the selected Event Handler microinstruction address on 50 CSADR Bus 20284.

To summarize, a number of Events may request service by JP 10114. Of these Events, all, some, or none, may be recognized. Only one Event Request, the highest priority Event Request, will be honored when JP 10114 enters State AB. Microinstruction control of CS 10110 will then transfer to that Event's Handler microinstruction sequence. A necessary condition for entering State AB is that an Event Request 55 has been made and recognized.

A microinstruction sequence "completes", "is completed", or reaches "completion" when CS 10110 exits State M1 while that microinstruction sequence is active. A microinstruction sequence may, as described above, be aborted in State M0 an indefinite number of times before, if ever, reaching completion.

60 A MEM 10112 reference "completes", "is completed", or reaches "completion" when requested data is returned to the specified destination, that is read from MEM 10112 to the requestor, or MEM 10112 accepts data to be written into MEM 10112.

"Trace Traps" are an inherent feature of microinstructions being executed. Trace Traps occur on every microinstruction of a given type (if not masked), for example during a sequence of microinstructions to perform a Name evaluate or resolve, and occur on each microinstruction of the sequence. In general, a 65 Trace Trap Event must be serviced before execution of the next microinstruction. Trace Traps are distinct

from Interrupts in that an Interrupt, described below, does not occur on execution of each microinstruction of a microinstruction sequence, but only on those microinstructions where certain other conditions must be considered.

"Interrupts" are the largest class of events in JP 10114. Occurrence of an Interrupt may not, in general, be predicted for a particular execution of a particular microinstruction in a particular instance. Interrupts may require service before execution of the next microinstruction, before execution of the current microinstruction can complete, or before beginning of the next SIN. An Interrupt may be unrelated to execution of any microinstruction, and is serviced before beginning of the next microinstruction.

A "Machine Check" is an Event that JP 10114 may not handle alone, or whose occurrence makes further actions by JP 10114 suspect. These events are captured in EVENT 20284 Registers and result in a request to DP 10118 to stop operation of JP 10114 for subsequent handling.

In summary, three major classes of Events in CS 10110 are Trace Traps, Interrupts, and Machine Checks. Each of these class of events will be described in further detail below, beginning with Trace Traps.

The State of all possible Trace Trap Event Requests, whether requesting or not requesting, is loaded into EVENT 20284 Registers at completion of State M1 and at completion of State AB. That is, since Trap Requests are a function of the currently executing microinstruction, the State of a Trap Request will be loaded into EVENT 20284 Trace Trap Registers at end of State M1 of each currently executing microinstruction. Similarly, if any Trap Requests are recognized, State AB will be entered at the end of the first clock cycle of the next following State M0 and their State loaded at end of the State AB.

Recognized, or unmasked, Trap Requests may be pushed onto RCWS 10358 as Pending Requests. Unrecognized, or masked, Trace Trap Requests may be pushed onto RCWS 10358 as Not Pending Requests and are subsequently disregarded. Subsequently, when a microinstruction sequence ends in a return to a calling microinstruction sequence, the Trace Trap Request bits in an RCWS 10358 may be used to generate Trace Trap Event Requests.

Upon exit from State AB, all Trace Trap Requests, except Micro-Break-Point and Microinstruction Trace Traps, described below, are loaded into corresponding EVENT 20284 Trace Trap Request Registers as not requesting. Micro-Break-Point and Microinstruction Trace Traps, are, in general, always latched as requesting at completion of State AB. Trace Traps may be explicitly masked by a Trace Mode Mask, an Indivisibility Mode Mask, and by a Trace Enable input, all generated by FUSITT 11012 as described below. Micro-Break-Point Trap may also be masked by clearing a Trace Enable bit in a Trace Enable field of certain microinstructions containing Trace Traps. In general, masking is effective from State M0 of the microinstruction which generates the mask, through completion of a microinstruction which clears the mask Trace Traps generated by a microinstruction which clears a mask are taken so as to abort a following microinstruction during its M0 State.

Referring to Fig. 245, CS 10110 state timing for a typical Trap Request, and generation of a microinstruction address to a corresponding Trace Trap Handler microinstruction sequence by EVENT 20284 is shown. Fig. 245 is drawn using the same conventions as described above with reference to Fig. 244A to 244Z. In Fig. 245, a microinstruction executing in States M0 and M1 causes a Trace Trap Request but does not generate an MR (Memory Reference) Trailer State. Trace Trap Request to EVENT 20284 is signaled by Time A. This Trace Trap Request is latched into EVENT 20284 Trace Trap Event Registers, and an Abort Request is provided to STATE 20294. At Time B, FU 10120 enters States AB and AR. The microinstruction address for a Handler microinstruction sequence of the highest priority Event present in EVENT 20284 is presented to FUSITT 11012 and execution of the addressed microinstruction sequence begins. At Time C, FU 10120 exits States AB and AR and enters State AB. State AB will be exited at end of the next 110 nanosecond clock cycle. Address of the selected Event Handler microinstruction sequence will remain on CSADR Bus 20204 for duration of State AB. At Time D, a pointer into RCWS 10358, described in a following description, is incremented, thereby effectively pushing the first microinstruction's return control word, that is the microinstruction executing at first State M0, onto RCWS 10358. First microinstruction of the Trace Trap Event Handler microinstruction sequence is provided by FUSITT 11012. Execution of Handler microinstruction sequence will begin at start of the third State M0 of the state timing sequence shown in Fig. 245. EVENT 20284's Trace Trap Register for this event is now latched in nonrequesting state and will remain so until transition out of second State M1 shown in Fig. 245. At this time, EVENT 20284 Registers will latch new Trap Requests. Finally, at Time E, Trace Trap Event Registers of EVENT 20284 are latched with new Trap Requests arising from execution of the microinstruction being executed in States M0 and M1 occurring between Times D and E. Traps due to the microinstruction that was executed in States M0 and M1 before Time A, but were not serviced, are requested again when the previously pushed RCW described above is returned from RCWS 10358 upon return from the Trace Trap Event Handler microinstruction sequence initiated at Time D. All Trace Trap Requests which have been serviced are explicitly cleared in RCWS 10358 RCWs by their Event Handler microinstruction sequences to prevent recurrence of those Trap Requests. Since Trace Trap Event Requests arising from reads or writes to MEM 10112 will recur if those requests are repeated, EVENT 20284 generates memory repeat Interrupts after all aborted MEM 10112 read and write requests to insure that these Traps will eventually be serviced. Event Handler microinstruction sequences for these read and write Trace Trap Events explicitly disable serviced Trace Trap Event Requests by clearing bits in the logical descriptor of the aborted memory read and write requests.

EP 0 067 556 B1

Having described overall structure and operation of Trace Trap Events, certain specific Trace Trap Events will be described in greater detail below. Trace Trap Events occurring in CS 10112 may include Name Trace Traps, SOP Trace Traps, Microinstruction Trace Traps, Micro-Break-Point Trace Traps, Logical Write Trace Traps, Logical Read Trace Traps, UID Read Trace Traps, and UID Write Trace Traps. These Trace Traps will be described below in the order named.

A Name Trace Trap is requested upon every microinstruction sequence that contains an evaluate or resolve of a Name syllable. Name Trace Traps are provided by decoding certain microinstruction fields of those microinstruction sequences. Name Trace Trap field is masked by either Trace Mask, Indivisibility Mask, or Trace Enable, as described above. All of these masks are set and cleared by microinstruction control signals provided during microinstruction sequences calling for resolves or evaluates of Name syllables.

A SOP Trace Trap may be requested whenever FU 10120 enters State FM (First Microinstruction of an SOP). SOP Trace Traps may be masked by Trace Mask, Indivisibility Mask, or Trace Trap Enable, again provided by microinstruction control outputs of FUSITT 11012. In general, the first microinstruction of such a microinstruction sequence interrupting such SOPs is not completed before a Trace Trap is taken.

Microinstruction Trace Traps may be requested upon completion of microinstructions which do not contain a Return Command, that is those microinstructions which do not return microinstruction control of CS 10110 to the calling microinstruction sequence. For microinstruction sequences containing Return Commands, state of microinstruction Trace Trap Request in a corresponding RCW is used. Every microinstruction for which a Microinstruction Trace Trap is not masked is aborted during State M0 of execution of that microinstruction. Microinstruction Trace Traps may be masked by Trace Mask, Indivisibility Mask, or Trace Enable from FUSITT 11012. A Micro-Break-Point Trap may be requested upon execution of microinstructions which do not contain Return Commands, but in which a Trace Enable bit in a microinstruction is asserted. A Micro-Break-Point Trap may be masked by Trace Mask, Indivisibility Mask, or Trace Enable. In addition, a Trace Enable bit of a microinstruction field in these microinstruction sequences controls recognition of Micro-Break-point Traps. Micro-Break-Point Traps are thereby requested whenever a microinstruction Trace Trap is requested, but have additional enabling conditions expressed in the microinstructions. Since only recognized Traps are pushed onto RCWS 10358 in a RCW, a Microinstruction Trace Trap and a Micro-Break-Point Trap having different request states may be present in RCWS 10358 concurrently.

Logical Write Trace Traps may be requested when enabled by a bit set in a logical descriptor during a microinstruction sequence submitting a write request to MEM 10112 and using logical descriptors to do so. Logical Write Trace Traps are recognized only if they occur during a state which will be immediately followed by State MR (Memory Reference Trailer). A Logical Write Trace Trap will result in the MEM 10112 write request being aborted. Logical Write Trace Traps may be masked by Trace Masks, Indivisibility Mask, or Trace Trap Enable. A further condition for recognition of a Logical Write Trace Trap is determined by the state of certain bits in a logical descriptor of the memory write request. Logical Write Trace Traps are, in general, not pushed onto RCWS 10358 as part of a RCW since aborted MEM 10112 requests are re-generated so that Logical Write Trace Traps may be repeated.

Logical Read Trace Traps are similar in all respects to the Logical Write Trace Traps, but occur during MEM 10112 read requests. Generation of Logical Read Trace Traps is controlled again in part by certain bits in logical descriptors of MEM 10112 read requests.

In certain implementations of CS 10110, UID Trace Traps may be requested when FU 10120 requests an MEM 10112 read operation based upon a UID address or pointer. UID Read Trace Traps are recognized if requested and there is, in general, no explicit masking of UID Read Trace Traps. Generation of UID Read Trace Traps is controlled by certain bits in MEM 10112 read request logical descriptors. UID Read Trace Trap Requests result in the MEM 10112 read requests being aborted and CS 10110 entering State AB. Handler microinstruction sequences for UID Read Trace Traps will, in general, reset the trapped enable bit in the MEM 10112 read request logical descriptor before re-issuing the MEM 10112 read request.

UID Write Trace Traps are similar to UID Read Trace Traps, and are controlled by bits in the logical descriptor in MEM 10112 write request based upon UID addresses or pointers.

Having described above structure and operation of Trace Trap Events, CS 10110 Interrupt Events will be described next below.

As previously described, Interrupts form the largest class of CS 10110 Events. Interrupts may be regarded as falling into one or more of several classes. First, Memory Reference Repeat Interrupts are those Interrupt Events associated, in general, with read and write requests to MEM 10112 in which a read or write request is submitted to MEM 10112, and an Interrupt Event results. That Interrupt Event is handled, and the MEM 10112 request repeated. Second, Deferred Service Interrupts are those Interrupts wherein CS 10110 defers service of an Interrupt until entry to a new SIN. Fourth, Microinstruction Service Interrupts occur when a currently executing microinstruction requires assistance of an Event Handler microinstruction sequence to be completed. Finally, Asynchronous Interrupt Events may occur at any time and must be serviced before CS 10110 may exit State M0 of the next microinstruction. These Interrupt Events will be described next below in the order named.

A Memory Reference Repeat Interrupt is requested, for example, if a microinstruction executes a command, and a corresponding RCW read from RCWS 10358 indicates that a memory reference was

aborted before entrance to the microinstruction sequence from which return was executed. This type of Interrupt Event occurs for all aborted memory references. If an event is honored, that is abort state is entered, for any event and there is a memory reference outstanding, not aborted, the memory reference completes before State AB is exited. No memory Repeat Interrupt Request will be written into the RCW written onto RCWS 10358. Conversely, if a memory reference is aborted, even if the event honored is not that event which aborted the memory reference, a Memory Repeat Interrupt Request will be written into a RCW pushed onto a RCWS 10358.

There are two state timing sequences for execution of Memory Repeat Interrupts. In the first case, there are no MEM 10112 references in the microinstruction executing a Return Command. In the second case, a microinstruction executing a Return Command executes a return and also makes a MEM 10112 reference. Referring to Fig. 246, a CS 10110 State Timing Diagram for the first case is shown. Fig. 246 is drawn using the same conventions as used in Fig. 244 and 245. As described above, in the first case a microinstruction executing a Return Command is executed in States M0 and M1 following Time D. An aborted MEM 10112 reference was made in States M0 and M1 preceding Time A. A MEM 10112 Reference Abort Request is made upon CS 10110's entry into State MR following Time A. Since a Memory Repeat Interrupt is requested only from a RCW provided by RCWS 10358, a Memory Repeat Interrupt is indicated only if a microinstruction executes a Return Command resulting in RCWS 10358 providing such an RCW. Therefore, a Memory Repeat Interrupt Request Register of EVENT 20284 is loaded with "not requesting" at this time. At Time B, CS 10110 enters State AB, State AR, and State MA. At this time, a Memory Reference Abort Request is asserted and written into an RCW when State AB is exited just before Time D. At Time D, CS 10110 exits State AR and State MA. As just described, CS 10110 will remain in State B until Time D. At Time D, Memory Reference Abort Request is written into RCWS 10358 as part of an RCW and, as described further below, various RCWS 10358 Stack Pointers are incremented to load that RCW into RCWS 10358. At this time, EVENT 20284's Interrupt Request Register receives "no request" as state of Memory Repeat Interrupt. First microinstruction of Memory Repeat Interrupt Handler microinstruction sequence is provided by FUSITT 11012. At Time E, the last microinstruction of the Memory Repeat Interrupt Handler microinstruction sequence is provided by FUSITT 11012 and a Return Command is decoded. RCWS 10358 Previous Stack Pointer, previously described, is selected to address RCWS 10358 to provide the previously written RCW as output to EVENT 20284's Memory Repeat Interrupt Event Register. At Time F, EVENT 20284's Memory Repeat Interrupt Register is loaded from output of RCWS 10358 and RCWS 10358's Stack Register Pointers are decremented. At this time, Memory Repeat Interrupt Request is made and, as described below, is written into the current Return Control Word, whether honored or not. JP 10114 then repeats the aborted MEM 10112 reference.

In the second case, a State Timing Sequence wherein the microinstruction executing a return also makes a MEM 10112 reference, CS 10110 State Timing is identical up to Time F. At Time F, MEM 10112 Repeat request is not recognized and the state of Memory Repeat Interrupt written into the current Return Control Word is "not requesting" unless a current MEM 10112 reference is aborted. The previous MEM 10112 Repeat Interrupt Request is disregarded as it is assumed that it is no longer required. Thus, there are two ways to avoid, or cancel a Memory Repeat Interrupt Request. First, that portion of a RCW receiving a MEM 10112 Repeat Interrupt Request may be rewritten as "not requesting". Second, an aborted MEM 10112 reference may be made in the same microinstruction that returns from a Handler servicing the aborted MEM 10112 reference.

Certain CS 10110 Events result in aborting a MEM 10112 read or write references and may result in repeat of MEM 10112 references. These events may include:

- (1) Logical read and write Traps and, in certain implementations of CS 10110, UID read and write Traps, previously discussed;
- (2) A PC 10234 miss;
- (3) Detection of a protection Violation by PC 10234;
- (4) A Page Crossing in a MEM 10112 read or write request;
- (5) A Long Address Translation, that is an ATU 10228 miss requiring JP 10114 to evaluate a logical descriptor to provide a corresponding physical descriptor;
- (6) Detection of a reset dirty bit flag from ATU 10228 upon a MEM 10112 write request as previously described;
- (7) An FU 10122 stack overflow;
- (8) An FU 10122 Illegal Dispatch;
- (9) A Name Trace Trap event as previously described;
- (10) A Store Back Exception, as will be described below;
- (11) EU 10122 Events resulting in aborting of a Store Back, that is a write request to MEM 10112 from EU 10122;
- (12) A read request to a non-accelerated Stack Frame, that is a Stack Frame presently residing in MEM 10112 rather than accelerated to JP 10114 Stack Mechanisms; and,
- (13) Conditional Branches in SIN sequences resulting outstanding MEM 10112 read reference from PREF 20260; and,

Of these Events, Logical Read and Write Traps, UID Read and Write Traps, and Name Trace Traps have been previously described. Other Events listed above will be described next below in further detail.

A PC 10234 Miss Interrupt may be requested upon a logical MEM 10112 reference, that is when a logical descriptor is provided as input to ATU 10228 and a protection state is not encached in PC 10234. PC 10234 will, as previously described, indicate that a corresponding PC 10234 entry is not present by providing a Event Protection Violation (EVENTPVIOL) output to EVENT 20284. PC 10234 will concurrently assert an Abort output (ABORT) to force CS 10110 into State AB and thus abort that MEM 10112 reference.

A Page Crossing MEM 10112 Reference Interrupt is requested if a logical MEM 10112 reference, that is a logical descriptor, specifies an operand residing on two logical pages of MEM 10112. An output of ATU 10228 will abort such MEM 10112 references by asserting an Abort output (ABORT).

A Protection Violation Interrupt is requested if a logical MEM 10112 reference does not possess proper access rights, a mode violation, or if that reference appears to refer to an illegal portion of that object, an extent violation. Again, PC 10234 will indicate occurrence of a Protection Violation Event, which may be disabled by a microinstruction control output of FUSITT 11012.

A Long Address Translation Event may be requested upon a logical MEM 10112 reference for which ATU 10228 does not have an encached entry. ATU 10228 will abort that MEM 10112 reference by asserting outputs ABORT and Long Address Translation Event (EVENTLAT).

A Dirty Bit Reset Event Interrupt may be requested when JP 10114 attempts to write to an MEM 10112 page having an encached entry in ATU 10228 whose dirty bit is not set. ATU 10228 will abort that MEM 10112 write request by asserting outputs ABORT and Write Long Address Translation Event (EVENTWLAT).

An FU 10120 User Stack Overflow Event may be requested if the distance between a Current Frame Pointer and a Bottom Frame Pointer, previously described with reference to CS 10110 Stack Mechanisms, is greater than a given value. As previously described, in CS 10110 this value is eight. A User Stack Overflow Event will continue to be requested until either Current Frame Pointer or Bottom Frame Pointer changes value so that the difference limit defined above is no longer violated. A User Stack Overflow Event may be masked by a Trace Mask, an Indivisibility Mask, or by enable outputs of a microinstruction from FUSITT 11012. A Handler microinstruction sequence for User Stack Overflow Events must be executed with one or more of these masks set to prevent recursion of these events. CS 10110 is defined to be running on Monitor Stack (MOS) 10370 when User Stack Overflow Events are masked. User Stack Overflow Events are not loaded into any of EVENT 20284's Event Registers, nor are these events written into a RCW to be written onto RCWS 10358.

Illegal EU 10122 Dispatch Events are requested by EUSDT 20266 if FU 10120 attempts to dispatch, or provide an initial microinstruction sequence address, to EU 10122 to a EUSITT address which is not accessible to a user's program. Illegal EU 10122 Dispatch Events are, in general, not masked. Illegal EU 10122 Dispatch Event Requests are cleared upon CS 10110 exits from State AB. The Handler microinstruction sequence for Illegal EU 10122 Dispatch Events should, in general, reset Illegal EU 10122 Dispatch Event entries in RCWs to prevent recursion of these events.

EU 10122 will indicate a Store Back Exception Event if any one of a number of exceptional conditions arise during arithmetic operations. These events are recognized when CS 10110 enters State SB and are ignored except during Store Back to MEM 10112 of EU 10122 results. These Events may be disabled by microinstruction output of FUSITT 11012 but are, in general, not masked. Store Back Exception Events may be written into RCWs, to be stored in RCWS 10358, and are cleared upon CS 10110's exit from State AB. Again, a Store Back Exception Event Handler microinstruction sequence should reset Store Back Exception Events written into RCWs to prevent recursion of these events.

As described above, the next major class of Interrupt Events are Deferred Service Interrupts. CS 10110 defers service of Deferred Service Interrupts until entry of a new SOP Deferred Service Interrupts which have been recognized will be serviced before completion of execution of the first microinstruction of that new SOP. Deferred Service Interrupts include Nonfatal MEM 10112 Errors, Interval Timer Overflows, and Interrupts from IOS 10116. These Interrupts will be described below, in the order named.

A Nonfatal MEM 10112 Interrupt is signaled by MEM 10112 upon occurrence of a correctable (single bit) MEM 10112 error. Nonfatal Memory Error Interrupts are recognized only during State M0 of the first microinstruction of an SOP. MEM 10112 will continue to assert Nonfatal Memory Error Interrupt until JP 10114 issues an acknowledgement to read MEM 10112's Error Log.

An Interval Timer Overflow Interrupt is indicated by TIMERS 20296 when, as described below, an Interval Timer increments to zero, thus indicating lapse of an allowed time limit for execution of an operation. Interval Timer Overflow Interrupts are recognized during State M0 of the first microinstruction of a SOP. TIMERS 20296 will continue to request such interrupts until cleared by a microinstruction output of FUSITT 11012.

IOS 10116 will indicate an IOS 10116 Interrupt to indicate that an inter-processor message from IOS 10116 to JP 10114 is pending. IOS 10116 will continue to assert an IOS 10116 Interrupt Request, which is stored in a register, until cleared by a microinstruction control output of FUSITT 11012. IOS 10116 Interrupts are recognized during State M0 of the first microinstruction of an SOP.

The next major class of CS 10110 events are Interrupts due to the requirement by microinstruction sequences to be serviced in order to complete execution. These Interrupts must be serviced before a microinstruction sequence may be completed. Microinstruction Service Interrupts include Illegal SOP Events, Microinstructions Not Present in FUSITT 11012 Events, an attempted parse of a hung INSTB 20262, underflow of an FU 10120 Stack, an NC 10226 Cache Miss, or an EU 10122 Stack Overflow. Each of these

events will be described below, in the order named.

An Illegal SOP Event is indicated by FUSDT 11010 to indicate that a current SOP Code is a Long Code, that is greater than eight bits, while the current dialect (S-Language) expects only Short Operation Codes, that is eight bit SOPs. An Illegal SOP Interrupt is not detected for unimplemented SOPs within the proper code length range. Illegal SOP Events are, in general, not masked. FUSDT 11010 continues to indicate an Illegal SOP Event until a new SOP is loaded into OPCODEREG 20268. Illegal SOP Events are recognized during the first microinstruction of an SOP, that is during State FM. Should a Handler microinstruction sequence for a higher priority event change contents of OPCODEREG 20268, a previous Illegal SOP Event will be indicated again when the aborted SOP is retried.

Absence of a Microinstruction in FUSITT 11012 is indicated by FUSITT 11012 asserting a Control Store Address Invalid (CSADVALID). This FUSITT 11012 output indicates that that particular microinstruction address points outside of FUSITT 11012's address space. Output of FUSITT 11012 in such event is not determined and parity checking, described below, of microinstruction output is inhibited. The Handler microinstruction sequence for these Events will load FUSITT 11012 address zero with the required microinstruction from MEM 10112, as previously described, and return to the original microinstruction sequence.

An attempted parse of a hung INSTB 20262 is indicated by INSTBWC 24110 when a parse operation is attempted, INSTB 20262 is empty, and PREF 20260 is not currently requesting SInS from MEM 10112. In general, these Events are not masked. If a higher priority Event is serviced, these Events are indicated again when the aborted microinstruction is retried if the original conditions still apply.

An FU 10120 Stack Underflow Event is requested when a current microinstruction references a Previous Stack Frame which is not in an accelerated stack, that is, the Current Stack Pointer equals Bottom Stack Pointer. FU 10120 Underflow Events are, in general, not masked and are requested again on a retry if the microinstruction is aborted and this event has not been serviced.

An NC 10226 Miss Interrupt occurs on a MEM 10112 read or write operation when a load or read of NC 10226 is attempted and there is no valid NC 10226 block corresponding to that Name syllable. An NC 10226 Miss Event does not result in a request for a Name evaluate or resolve. In general, these Events are not masked and result in a request being issued again if the microinstruction resulting in that Event is retried and has not been serviced.

An EU 10122 Stack Overflow Event is requested from EU 10122 to indicate that EU 10122 is currently already servicing at least one level of Interrupt an FU 10122 is requesting another. As will be described in a following description of EU 10122, EU 10122 contains a one level deep stack for handling of Interrupts. EU 10122 Stack Overflow Events are enabled during State NW. All previously pending events will have been serviced before EU 10122 Stack Overflow Event requests are recognized. These Events will be serviced immediately upon entry into a following State M0, being the highest priority interrupt event. EU 10122 Stack Overflow Events may, in general, not be masked and once recognized are the next honored event.

Finally, the third major class of CS 10110 Interrupt Events are Asynchronous Events. Asynchronous Events must, in general, be serviced before exiting State M0 of a microinstruction after they are recognized. Asynchronous Events include Fatal Memory Error Events, AC Power Failure Events, Egg Timer Overflow Events, and EU 10122 Stack Underflow Events. CS 10110 Egg Timer is a part of TIMERS 20296 and will be discussed as part of TIMERS 20296. These events will be described below, in the order referred to.

Fatal MEM 10112 Error Events are requested by MEM 10112 by assertion of control signal output PMODI, previously described, when last data read from MEM 10112 contains a noncorrectable error. Fatal MEM 10112 Error Events are recognized on first State M0 after occurrence. Fatal MEM 10112 Error Events are stored in an EVENT 20284 Event Register and are cleared upon entry into its service microinstruction sequence. In general, Fatal MEM 10112 Error Events may not be masked.

AC Power Failure Events are indicated by DP 10118 by assertion of output signal ACFAAIL when DP 10118 detects a failure of power to CS 10110. Recognition of AC Power Failure Events is disabled upon entry to AC Power Failure Event Handler microinstruction sequence. No further AC Power Failure Events will be recognized until DP 10118 reinitiates JP 10114 operation.

As will be described further below, FUCTL 20214's Egg Timer is a part of TIMERS 20296. Egg Timer Overflow Events are indicated by TIMERS 20296 whenever TIMERS 20296's Egg Timer indicates overflow of Egg Timer Counter. Egg Timer Overflow Events may be masked as described in a following description.

Finally, EU 10122 Stack Underflow Events are signaled by EU 10122 when directed to read a word from EU 10122 Stack Mechanism and there is no accelerated stack frame present. EU 10122 will continue to assert this Event Interrupt until acknowledged by JP 10114 by initiation of a Handler microinstruction sequence.

The above descriptions of CS 10110 events have stated that recognition of certain of those Events may be masked, that is inhibited to allow recognition of other Events having higher priority. Certain of these masking operations were briefly described in the above descriptions and will be described in further detail next below. In general, recognition of Events may be masked in five ways, four of which are properly designated as masks. These four masks are generated by microinstruction control from FUSITT 11012 and include Asynchronous Masks for, in general, Asynchronous Events. Monitor Masks are utilized for those CS 10110 operations being performed on Monitor Stack (MOS) 10370, as previously described with reference to CS 10110 Stack Mechanisms. Trace Mask is utilized with reference to Trace Trap Events. Indivisible Mask

is generated or provided by FUSITT 11012 as an integral or indivisible part of certain microinstructions and allow recognition of certain selected events during certain single microinstructions. Certain other Events, for example Logical Read and Write Traps and UID Read and Write Traps, are recognized or masked by flag bits in logical descriptors associated with those operations. Finally, certain microinstructions result in FUSITT 11012 providing microinstruction control outputs enabling or inhibiting recognition of certain events, but differ from Indivisible Masks in not being associated with single particular microinstructions.

Referring to Fig. 247, the relative priority level and applicable masks of certain CS 10110 Events are depicted therein in three vertical columns. Information regarding priority and masking of particular Events is shown in horizontal entries, each comprising an entry in each of these three vertical columns. Left hand column, titled Priority Level, states relative priority of each Event entry. Second column, titled EVENT, specifies which Event is referred to in that table entry. A particular Event will yield priority to all higher priority Events and will take precedence over all lower priority Events. Fig. 247's third column, titled Masked By, specifies for each entry which masks may be used to mask the corresponding Event. A indicates use of Asynchronous Masks, M use of Monitor Mask, T use of Trace Trap Mask, and I represents that Indivisible Mask may be used. DES indicates that an Event is enabled or masked by flag bits of logical descriptors, while MCWD indicates that a particular Event may be masked by microinstruction control signal outputs provided by FUSITT 11012. NONE indicates that a particular Event may, in general, not be masked.

The final major class of CS 10110 event was described above as Machine Check Events. In general, if any of these Events are detected by logic gating in EVENT 20284, EVENT 20284 will provide a Check Machine signal to DP 10118. DP 10118 will then stop operation of JP 10114 and Machine Check Event Handler microinstruction sequences will be initiated. Among these Machine Check Events are wherein FU 10120 is attempting to store back an EU 10122 result to MEM 10112 and EU 10122 signals a parity error in EU 10122's Control Store. These events are stored in EVENT 20284 Event Registers and recognized when FU 10120 enters State AB. EU 10122 will have previously ceased operation until a corrective microinstruction sequence may be initiated. The same Event will occur if FU 10120 attempts to use an EU 10122 arithmetic operation result or test operation result having a parity error in EU 10122's Control Store. Should MOS 10370 overflow or underflow, this event will be detected, FU 10120 operations stopped, and corrective microinstruction sequences initiated. MOS 10370 overflow or underflow occurs whenever a previous MOS 10370 Stack Frame is referenced, whenever MOS 10370 Stack Pointer equals MOS 10370 Bottom Stack Pointer, or the difference between MOS 10370 Current and Bottom Stack Pointers is greater than sixteen. Underflows result in a transfer of operation to MIS 10368, while overflows are handled by DP 10118. Finally, a Machine Check Event will be requested when a parity error is detected in a microinstruction currently being provided by FUSITT 11012 during State M0 of that microinstruction.

Having described general operation of EVENT 20284, the structure and operation of EVENT 20284 will be described briefly next below.

Referring to Fig. 248, a partial block diagram of EVENT 20284 is shown. EVENT 20284 includes Event Detector (EDET) 24810, Event Mask and Register Circuitry (EMR) 24812, and Event Handler Selection Logic (EHS) 24814. EDET 24810 is comprised of random logic gating and, as previously described, receives inputs representing event conditions from other portions of CS 10110's circuitry. EDET 24810 detects occurrences of CS 10110 operating conditions indicating that Events have occurred and provides outputs to EMR 24812 indicating what Events are requested.

EMR 24812 includes a set of registers, for example SN74S194s, comprising EVENT 20284's Event Registers. These registers are enabled by mask inputs, described momentarily, to enable masking of those Events which are latched in EVENT 20284's Event Registers. Certain Events, as previously described, are not latched and logic gating having mask enable inputs is provided to enable masking of those events which are not latched. EMR 24812 mask inputs are Asynchronous, Monitor, Trace Trap, and Indivisible Masks, respectively AMSK, MMSK, TMSK, and ISMK, provided from FUSITT 11012. Mask inputs derived from FUSITT 11012 microinstruction outputs (mWRD) are provided from microinstruction control outputs of FUSITT 11012. EMR 24812 provides outputs representing mask and unmask events which have been requested to EHS 24814.

EHS 24814 is comprised of logic gating detecting which of EHS 24814's unmasked Event Requests is of highest priority. EHS 24814 selects the highest priority unmasked Event Request input and provides a corresponding Event Handler microinstruction address to EVNTGT 24310 through ADRA Bus 24322. These address outputs of EHS 24814 are five bit addresses selecting the initial microinstruction of the Event Handler microinstruction sequence of the current highest priority unmasked Event. As previously described with reference to NASMUX 24312, certain inputs of ENTGT 24310 are hard-wired to provide a full fifteen bit address output from EVNTGT 24310. EVENT 20284 also provides, from EHS 24814, an Event Enable Select (EES) output to SITNAS 20286 to enable EVNTGT 24310 to provide microinstruction addresses to CSADR Bus 20204 when EVENT 20284 must provide a microinstruction address for handling of a current Event.

Having described the structure and operation of FUCTL 20214's circuitry providing microinstruction addresses to FUSITT 11012, FUSITT 11012 will be described next below.

c.c.c. Fetch Unit S-Interpreter Table 11012 (Fig. 249)

Referring to Fig. 249, a partial block diagram of FUSITT 11012 is shown. Address (ADR) and Data

(DATA) inputs of Micro-Instruction Control Store (mCS) 24910 are connected, respectively, from CSADR Bus 20204 through Address Driver (ADRDRV) 24912 and from JPD Bus 10142 through Data Driver (DDRV) 24194. mCS 24910 comprises a memory for storing sequences of microinstructions currently being utilized by CS 10110. mCS 24910 is an 8K (8192) word by 80 bit wide memory. That is, mCS 24910 may contain, for example, up to, 8192 80 bit wide microinstructions. Microinstructions to be written into mCS 24910 are provided, as previously described, to mCS 24910 DATA input from JPD Bus 10142 through DDRV 24914. Addresses of microinstructions to be written into or read from mCS 24910 are provided to mCS 24910 ADR input from CSADR Bus 20204 through ADRDRV 24912. ADRDRV 24912 and DDRV 24914 are buffer drivers comprised, for example, of SN74S240s and SN74S244s.

Also connected from output of ADRDRV 24912 is input of Nonpresent Micro-Instruction Logic (NPmIS) 24916. NPmIS 24916 is comprised of logic gating monitoring read addresses provided to mCS 24910. When a microinstruction read address present on CSADR Bus 20204 refers to an address location not within mCS 24910's address space, that is of a non-present microinstruction, NPmIS 24916 generates an Event Request output indicating this occurrence. As previously described FUCTL 20214 will then call, and execute, microinstructions so addressed from MEM 10112.

As indicated in Fig. 249, mCS 24910 provides three sets of outputs. These outputs are Direct Output (DO), Direct Decoded Output (DDO), and Buffered Decode Output (BDO). In general, control information within a particular microinstruction word is used on next clock cycle after the address of that particular microinstruction word has been provided to mCS 24910 ADR input. That is, during a first clock cycle a microinstruction's address is provided to mCS 24910 ADR input. That selected microinstruction appears upon mCS 24910's DO, DDO, BDO outputs during that clock cycle and are used, after decoding, during next clock cycle. Outputs DO, DDO, BDO differ in delay time before decoded microinstruction outputs are available for use.

mCS 24910 DO output provides certain bits of microinstruction words directly to particular destinations, or users, through Direct Output Buffer (DOB) 24918. These microinstructions bits are latched and decoded at their destinations as required. DOB 24918 may be comprised, for example, of SN74S04s.

mCS 24910's DDO output provides decoded microinstruction control outputs for functions requiring the presence of fully decoded control signals at the start of the clock cycle in which those decoded control signals are utilized. As shown in Fig. 249, mCS 24910's DDO output is connected to input of Direct Decode Logic (DDL) 24920. DDL 24920 is comprised of logic gating for decoding certain microinstruction word bits during same clock cycle in which those bits are provided by mCS 24910's DDO. These microinstruction bits are provided, as described above, during the same clock cycle in which a corresponding address is provided to mCS 24910's ADR input. During this clock cycle, DDL 24920 decodes mCS 24910's DDO microinstruction bits to provide fully decoded outputs by end of this clock cycle. Outputs of DDL 24920 are connected to inputs of Direct Decode Register (DDR) 24922. DDR 24922 is a register comprised, for example, of SN74S374s. DDL 24920's fully decoded outputs are loaded into DDR 24922 at the end of the clock cycle during which, as just described, an address is provided to mCS 24910's ADR input and mCS 24910's corresponding DDO output is decoded by DDL 24920. Fully decoded microinstruction control outputs corresponding to mCS 24910's DDO outputs are thereby available at start of the second clock cycle. Microinstruction control outputs of DDR 24922 are thereby available to FU 10120 at start of the second clock cycle for those FU 10120 operations requiring immediate, that is undelayed, microinstruction control signal outputs from FUSITT 11012.

Finally, mCS 24910's BDO is provided for those FU 10120 operations not requiring microinstruction control signals immediately at the start of the second clock cycle. As shown in Fig. 249, mCS 24910's BDO is connected to inputs of Buffered Decode Register (BDR) 24924. Microinstruction word output bits from mCS 24910's BDO are provided to inputs of BDR 24924 during the clock cycle in which a corresponding address is provided to mCS 24910's ADR input. mCS 24910's BDO outputs are loaded into BDR 24924 at end of this clock cycle. BDR 24924's outputs are connected to inputs of Buffered Decode Logic (BDL) 24926. BDL 24926 is comprised of logic gating for decoding outputs of BDR 24924. BDL 24926 thereby provides decoded microinstruction control outputs to FU 10120 at some delayed time after start of the second clock cycle. Microinstruction control outputs from BDL 24926 are thereby delayed in time from the appearance of microinstruction control outputs of DDR 24922 but, as BDR 24924 stores microinstruction word bits rather than decoded microinstruction word bits, BDR 24924 is required to store proportionately fewer bits than DDR 24922.

Finally, as shown in Fig. 249 outputs of DDR 24922 and BDR 24924, are connected to inputs of Microinstruction Word Parity Checker (mWPC) 24928. mWPC 24928 is comprised of logic gating for checking parity of outputs of DDR 24922 and BDR 24924. A failure in parity of either output of DDR 24922 and BDR 24924 indicates a possible error in microinstruction output from mCS 24910. When such an error is detected by mWPC 24928, mWPC 24928 generates a corresponding Microinstruction Word Parity Error (mWPE).

d.d. CS 10110 Internal Mechanism Control

Associated with SR's 10362, the stack mechanism area of GRF 10354, are two CS 10110 control structures primarily associated with operation of CS 10110's internal mechanisms. A first of these referred to as Machine Control Block, describes current execution environment of JP 10114 microprograms, that is,

EP 0 067 556 B1

JP 10114 microinstruction sequences. Machine Control Block is comprised of two information words residing in MCW1 20290 and MCW0 20292. These Machine Control Words contain all control state information necessary to execute JP 10114's current microprogram. Second control structure is a portion of RCWS 10358, which as previously described parallels the structure of SR's 10362. Each register frame on MIS 10368 or MOS 10370 has, with exception of Top (Current) Register Frame, associated with it a Return Control Word (RCW) residing in RCWS 10358. RCWs are created when MIS 10362 or MOS 10370 register frames are pushed, that is moved onto MIS 10368 or MOS 10370 due to creation of a new Current Register Frame. A current RCW does not exist in a present embodiment of CS 10110.

RCWS 10358 will be described first next below, followed by Machine Control Block.

a.a.a. Return Control Word Stack 10358 (Fig. 251)

Referring to Fig. 251, a diagramic representation of a RCWS 10358 RCW is shown. As previously described, RCWS 10358 RCWs contain information necessary to reinitiate or continue execution of a microinstruction sequence if execution of that sequence has been discontinued.

Execution of a microinstruction sequence may be discontinued due to a requirement to service a CS 10110 Event, as described above, or if that microinstruction sequence has called for execution of another microinstruction sequence, as in a Branch or Case Operation.

As shown in Fig. 251, each RCW may contain, for example, 32 bits of information. RCW Bits 16 to 31 inclusive are primarily concerned with storing current microinstruction address of microinstruction sequences which have been discontinued, as described above. Bits 17 to 31 inclusive contain microinstruction sequence return address. Return address is, as previously described, address of the microinstruction currently being executed of a microinstruction sequence whose execution has been discontinued. When JP 10114 returns from servicing of an Event or execution of a called microinstruction sequence, return address is provided from RCWS 10358 to SITNAS 20286 and through CSADR Bus 20204 to FUSITT 11012 as next microinstruction address to resume execution of that microinstruction sequence. Bit 16 of an RCW contains a state bit indicating whether the particular microinstruction referred to by return address field is the first microinstruction of a particular SOP. That is, Bit 16 of an RCW stores CS 10110 State FM.

Bits 8 to 15 inclusive of an RCW contain information pertaining to current condition code of JP 10114 and to pending Interrupt Requests. In particular, Bit 8 contains a condition code bit which, as previously described indicates whether a particular test condition has been met. RCW Bit 8 is thereby, as previously described, a means by which JP 10114 may pass results of a particular test from one microinstruction sequence to another Bits 9 to 15 inclusive of an RCW contain information regarding currently pending Interrupts. These Interrupts have been previously discussed, in general, with reference to EVENT 20284. In particular, RCW Bit 9 contains pending state of Illegal EU 10122 Dispatch Interrupt Requests; RCW Bit 10 contains pending state of Name Trace Trap Request; RCW Bit 11 contains pending state of Store Back Interrupt Request; RCW Bit 12 contains pending state of Memory Repeat Interrupt Request; RCW Bit 13 contains pending state of SOP Trace Trap Request; RCW Bit 14 contains pending state of Microtrace Trap Request; and, RCW Bit 15 contains pending state of Micro-Break Point Trap Request. Interrupt Handling microinstruction sequence which require use of CS 10110 mechanisms containing information regarding pending Interrupts must, in general, save and store that information. This save and restore operation is accomplished by use of Bits 9 to 15 of RCWS 10358's RCWs. Upon entry to an Interrupt Handling microinstruction sequence, these bit flags are set to indicate Interrupts which were outstanding at time of entry to that microinstruction sequence. Because these bits are used to initiate Interrupt Request upon returns, pending Interrupts may be cancelled by resetting appropriate bits of Bits 9 to 15 upon return. This capability may be used to implement Microinstruction Trace Traps, previously described.

As indicated in Fig. 251, RCW Bits 0 to 7 are not utilized in a present embodiment of CS 10110. RCW bits 0 to 7 are not implemented in a present embodiment of CS 10110 but are reserved for future use.

As previously described, RCWs may be written into or read from RCWS 10358 from JPD Bus 10142. This allows contents of RCWS 10358 to be initially written as desired, or read from RCWS 10358 to MEM 10112 and subsequently restored as required for swapping of processes in CS 10110.

b.b.b. Machine Control Block (Fig. 252)

As described above, FUCTL 20214's Machine Control Block is comprised of a Machine Control Word 1 (MCW1) and a Machine Control Word 0 (MCW0). MCW1 and MCW0 reside, respectively, in Registers MCW1 20290 and MCW0 20292. MCW1 and MCW0 described the current execution environment of FUCTL 20214's current microprogram, that is the microinstruction sequence currently being executed by JP 10114.

Referring to Fig. 252, diagramic representations of MCW0 and MCW1 are shown. As indicated therein, MCW0 and MCW1 may each contain, for example, 32 bits of information regarding current microprogram execution environment.

Referring to MCW0, MCW0 includes 6 execution environment subfields. Bits 0 to 3 inclusive contain a Top Of Stack Counter (TOSCNT) subfield which is a pointer to Current Frame of accelerated Microstack (MIS) 10368. TOSCNT field is initially set to point to Frame 1 of MIS 10368. Bits 4 to 7 inclusive comprise a Top of Stack -1 Counter (TOS-1CT) subfield which is a pointer to Previous Frame of accelerated MIS 10368, that is to the MIS 10368 frame proceeding that pointed by TOSCNT subfield. TOSCNT subfield is initially

set to Frame 0 of MIS 10368. Bits 8 to 11 inclusive comprise a Bottom of Stack Counter (BOSCNT) subfield which is a pointer to Bottom Frame of accelerated MIS 10368. BOSCNT subfield is initially set to point to Frame 1 of MIS 10368. TOSCNT, TOS-1CNT, and BOSCNT subfields of MCW0 may be read, written, incremented and decremented under microprogram control as frames are transferred between MIS 10368 and a SS 10336.

Bits 17 to 23 inclusive and Bits 24 to 31 inclusive of MCW0 comprise, respectively, Page Number Register (PNREG) and Repeat Counter (REPCTR) subfields which, together, comprise a microinstruction address pointing to a microinstruction currently being written into FUSITT 11012.

Bits 12 to 15 inclusive of MCW0 comprise an Egg Timer (EGGT) subfield which will be described further below with respect to TIMERS 20296. Bit 16 of MCW0 is not utilized in a present embodiment of CS 10110.

Referring to MCW1, MCW1 is comprised of four subfields. Of the 32 bits comprising MCW1, Bits 0 to 15 inclusive and Bits 24 and 25 are not utilized in a present embodiment of CS 10110. Bit 16 is comprised of a Condition Code (CC) subfield indicating results of certain test conditions in JP 10114. As previously described CC subfield is automatically saved and restored in RCWS 10358 RCW's.

Bits 17 to 19 inclusive of RCW1 comprise an Interrupt Mask (IM) subfield. The three bits of IM subfield are utilized to indicate a hierarchy of non-interruptible JP 10114 microinstruction control operating states. That is, a three bit code stored therein indicates relative power to interrupt between three otherwise noninterruptible JP 10114 operating states. Bits 20 to 23 inclusive comprise an Interrupt Request (IR) subfield which indicate Interrupt Request. These Interrupt Requests may include, for example, Egg Timer Overflow, Interval Timer Overflow, or Non-Fatal Memory Error, as have been previously described. Finally, Bits 26 to 31 inclusive comprise a Trace Trap Enable (TTR) subfield indicating which Trace Trap Events, previously described, are currently enabled. These enables may include Name Trace Enable, Logical Retrace Enable, Logical Write Trace Enable, SOP Trace Enable, Microinstruction Enable, and Microinstruction Break point Enable.

MCW0 and MCW1 has been described above as if residing in registers having individual, discrete existence, that is MCW1 20290 and MCW0 20292. In a present embodiment of CS 10110, MCW1 20290 and MCW0 20292 do not exist as a unified, discrete register structure but are instead comprised of individual registers having physical existence in other portions of FUCTL 20214. MCW1 20290 and MCW0 20292, and MCW1 and MCW0, have been so described to more distinctly represent the structure of information contained therein. In addition, this approach has been utilized to illustrate the manner by which current JP 10114 execution state may be controlled and monitored through JPD Bus 10142. As indicated in Fig. 202, MCW1 20290 and MCW0 20292 have outputs connected to JPD Bus 10142, thus allowing current execution state of JP 10114 to be read out of FUCTL 20214. Individual bits or subfields of MCW0 and MCW1 may, as previously described, be written by microinstruction control provided by FUSITT 11012. In a present physical embodiment of CS 10110, those registers of MCW0 20292 containing subfields TOSCNT, TOS-1CNT, and BOSCNT reside in RAG 20288. Those portions of MCW0 20292 containing subfield EGGT reside in TIMERS 20296. MCW0 20292 registers contain PNREG and REPCTR subfields are physically comprised of REPCTR 20280 and PNREG 20282. In MCW1 20290, CC subfield exists as output of FUCTL 20214 test circuits. Those MCW1 20290 registers containing IM, IR, and TTE subfields reside within EVENT 20284.

Having described FUCTL 20214 structure and operation as regards RCWS 10358, MCW1 20290 and MCW0 20292, FUCTL 20214, RAG 20288 will be described next below.

c.c.c. Register Address Generator 20228 (Fig. 253)

Referring to Fig. 253, a partial block diagram of RAG 20228, together with diagrammatic representation of GRF 10354, BIAS 20246 and RCWS 10358, is shown. As previously described, JP 10114 register and stack mechanisms include General Register File (GRF) 10354. BIAS 20246, and RCWS 10358. GRF 10354 is, in a present embodiment of CS 10110, a 256 word by 92 bit wide array of registers. GRF 10354 is divided horizontally to provide Global Registers (GRs) 10360 and Stack Registers (SRs) 10362, each of which contains 128 of GRF 10354's 256 registers. GRF 10354, that is both GRs 10360 and SRs 10362, is divided vertically into three vertical sections designated as AONGRF 20232, OFFGRF 20234, and LENGRF 20236. AONGRF 20232, OFFGRF 20234, and LENGRF 20236 are, respectively, 28 bits, 32 bits, and 32 bits wide. GRs 10360 is utilized as an array of 128 individual registers, each register containing one 92 bit word. SRs 10362 is structured and utilized as an array of 16 register frames wherein each frame contains eight registers and each register contains one 92 bit wide word. Eight of SR 10362's frames are utilized as Microstack (MIS) 10362 and the remaining eight of SR 10362's frames are utilized as Monitor Stack (MOS) 10370. For addressing purposes only, as described further below, GRs 10360 is regarded as being structured in the same manner as SRs 10362, that is as 16 frames of eight registers each.

BIAS 20246, as previously described, is a register array within BIAS 20246. BIAS 20246 contains 128 six bit wide registers, or words, and operates in parallel with and is addressed in parallel with SR 10362 portion of GRF 10354. RCWS 10358 is, as previously described, an array of 16 registers, or words, wherein each register contains one 32 bit RCW. RCWS 10358 is structured and operates in parallel with SRs 10362 with each RCWS 10358 register corresponding to a SR 10362 frame of eight registers. As described below, RCWS 10358 is addressed in parallel with SR 10362's frames.

Source and Destination Register Addresses (SDAR) for selecting a GRF 10354 register to be,

respectively, read from or written to are provided by RAG 20288. As described above BIAS 20246 operates and is addressed in parallel with SR 10362 portion of GRF 10354, that is parallel with SRs 10362. BIAS 20246 registers are thereby connected to and in parallel with address inputs of SRs 10362 and are addressed concurrently with GRs 10360. Registers RCWS 10358 also operate and are addressed in parallel with SRs 10362. Address inputs of RCWS 10358's registers are thereby connected in parallel with address inputs of SR 10362's registers.

RAG 20288's address inputs to GRF 10354, and to BIAS 20246 and RCWS 10358, may select registers therein to be either source registers, that is registers providing data, or destination registers, that is registers receiving data. RAG 20288's address outputs are designated as output Source and Destination Register Address (SDADR) of RAG 20288. RAG 20288's SDADR output is connected to address input of register comprising GRF 10354, BIAS 20246, and RCWS 10358. As described above, SRs 10362 are structured as 16 frames of 8 registers per frame and RCWS 10358 is structured as a corresponding 16 frames of one register per frame. GRF 10354 and BIAS 20246 are structured and utilized as single registers but, for addressing purposes, are regarded as being comprised of 16 frames of 8 registers per frame. Each SDADR output of RAG 20288 is an 8 bit word wherein the most significant bit indicates whether the addressed register, either a Source or a Destination Register, reside in GRs 10360 or within SRs 10362, BIAS 20246, and RCWS 10358. The four next most significant bits comprise a frame select field for selecting one of 16 frames within GRs 10360 or within SRs 10362, BIAS 20246, and RCWS 10358. The three least significant bits comprise a register select field selecting a particular register within the frame selected by frame select field.

Within a single system clock cycle, SDADR output of RAG 20288 may select a source register and data may be read from that source register, or SDADR output may select a destination register and data may be written into that destination register. As previously described, each JP 10114 microinstruction requires a minimum of two-system clock cycles for execution, that is at first clock cycle in State MO and a second clock cycle in State M1. During a single microinstruction therefore, a source register may be selected and data read from that source register, and a destination register selected and data written into that destination register. Certain operations, however, may require more than one microinstruction for execution. For example, a read-modify-write operation wherein data is read from a particular register, modified, and written back into that register may require two or more microinstructions for execution.

Referring first to RAG 20288 structure, RAG 20288 includes MISPR 10356. MISPR 10356 includes Top Of Stack Counter (TOSCNT) 25310, Top Of Stack-1 Counter (TOS-1CNT) 25312, and Bottom Of Stack Counter (BOSCNT) 25314. Contents of TOSCNT 25310, TOS-1CNT 25312 and BOSCNT 25314 are respectively, pointers to Current, Previous, and Bottom frames of SRs 10362, that is, to MIS 10368. As will be described below, these pointers are also utilized to address MOS 10370. TOSCNT 25310, TOS-1CNT 25312, and BOSCNT 25314 are each four bit binary counters comprised, for example, of SN74S163s.

Data inputs of TOSCNT 25310 to BOSCNT 25314 are connected from JPD Bus 10142. Control inputs of TOSCNT 15310 to BOSCNT 25314 are connected from microinstruction control outputs of FUSITT 11012. Data outputs of TOSCNT 25310 to BOSCNT 25314 are connected to data inputs of Source Register Address Multiplexer (SRCADR) 25316 and to data inputs of Destination Register Address Multiplexer (DSTADR) 25318. Data outputs of TOSCNT 25310 and BOSCNT 25314 are connected to inputs of Stack Event Monitor Logic (SEM) 25320.

Source and destination frame addresses are selected, as will be described further below, by SRCADR 25316 and DSTADR 25318 respectively. In addition to data inputs from TOSCNT 25310 and BOSCNT 25314, data inputs of SRCADR 25316 and DSTADR 25318 are connected from microinstruction word CONEXT subfield output from FUSITT 11012. Control inputs of SRCADR 25316 and DSTADR 25318 are connected from, respectively, microinstruction word RS and RD subfield outputs from FUSITT 11012. Source Frame Address Field (SRCFADR) output of SRCADR 25316 and Destination Frame Address Field (DSTFADR) output of DSTADR 25318 are connected to inputs of Source and Destination Register Address Multiplexer (SDADMUX) 25322. SRCFADR and DSTFADR comprise frame select fields of RAG 20288, SDADR outputs for, respectively, source and destination registers.

In addition to SRCFADR and DSTFADR outputs of SRCADR 25316 and DSTADR 25318, SDADMUX 25322 receives microinstruction word SRC and DST subfield inputs from microinstruction outputs of FUSITT 11012. As previously described, SRC subfield is a 3 bit number designating a source register, that is, a source register within a frame selected by SRCFADR. DST is similarly a 3 bit number selecting a destination register within a frame indicated by DSTFADR. SRC subfield input to SDADMUX 25322 is concatenated with SRCADR 25316 to respectively comprise, as described above, register and frame fields of a source register SDADR output of SDADMUX 25322. Similarly, DST subfield is concatenated with DSTFADR output of DSTADR 25318 to comprise, respectively, register and frame subfields of a destination register SDADR output of SDADMUX 25322. Selection between source and destination register address inputs to SDADMUX 25322, to generate a corresponding source or destination register SDADR output of SDADMUX 25322 is controlled by microinstruction control inputs (not shown for clarity of presentation) connected to control inputs of SDADMUX 25322. RDWS 25324 is a PROM decoding MD field from microinstruction words during reads from MEM 10112 and provides register select field of destination register address and selects one of the pointers as frame select field.

An Event output of SEM 25320 is connected to an input of EVENT 20284, previously described.

EP 0 067 556 B1

SRCADR 25316, DSTADR 25318, and SDADMUX 25322, as will be described further below, operate as multiplexers and may be comprised, for example, of SN74S153s.

Having described structure and organization of GRF 10354, BIAS 20246, and RCWS 10358, and structure of RAG 20288, operation of RAG 20288 to generate Source of Destination Register Address outputs SDADR will be described next below. Addressing of JP 10114's stack mechanism, comprising SRs 10362 and RCWS 10358, will be described first, followed by addressing of GRs 10360 and BIAS 20246.

SR 10362 portion of GRF 10354, RCWS 10358, and BIAS 20246 are addressed by Current, Previous, and Bottom Frame Pointers contained, respectively, in TOSCNT 25310, TOS-1CNT 25312, and BOSCNT 25314. Current, Previous, and Bottom Pointers comprise frame select fields of SDADMUX 25322. As previously described, Current, Previous and Bottom Pointer outputs of TOSCNT 25310 to BOSCNT 25314 are provided as inputs of SRCADR 25316 and DSTADR 25318. Microinstruction word RS subfield to control input of SRCADR 25316 selects either Current, Previous or Bottom Pointer input of SRCADR 25316 to comprise SRCFADR output of SRCADR 25316, that is to be frame select field of source register address. Similarly, microinstruction word RD subfield to control input of DSTADR 25318 concurrently selects either Current, Previous, or Bottom Pointer inputs of DSTADR 25318 to comprise DSTADR 25318's concurrently selects either Current, Previous, or Bottom Pointer inputs of DSTADR 25318 to comprise DSTADR 25318's DSTFADR output, that is frame select field of destination register address. As described above, SRCFADR and DSTFADR are provided as inputs to SDADMUX 25322. Microinstruction word SRC and DST subfield inputs to SDADMUX 25322 concurrently determine, respectively, source and destination registers within source and destination frames specified by SRCFADR and DSTFADR. SDADMUX 25322 then, operating under microinstruction control, selects either SRCFADR and SRC to comprise SDADR output to SR 10362 as a source register address or selects DSTFADR and DST as SDADR output specifying a destination register address. By microinstruction control of SRCADR 25316, DSTADR 25318, and SDADMUX 25322, a CS 10110 microprogram may select a source frame and register within SR 10362 and simultaneously specify a possible different destination frame and register within SR 10362. All possible combinations of source frame and register and destination frame and register in GRs 10360, SRs 10362, BIAS 20246, and RCWS 10358 are valid.

Control of SRCADR 25316, DSTADR 25318, and SDADMUX 25322 in addressing SR 10362 portion of GRF 10354, and RCWS 10358, is controlled, in part, by current CS 10110 state. Pertinent CS 10110 operating states, previously described, are State M1 and State RW. When CS 10110 is in neither State RW nor State M1, SR 10362 is addressed through SRCADR 25316 and microinstruction word SRC subfield, that is SR 10362 and RCWS 10358 are provided with source register addresses when CS 10110 is in neither RW nor M1 States. When CS 10110 enters State M1, SR 10362 and RCWS 10358 is addressed through DSTADR 25318 and by microinstruction word DST subfield. That is, SR 10362 and RCWS 10358 are provided with destination register addresses during State M1. Similarly, SR 10362 and RCWS 10358 are provided with destination register addresses when CS 10110 is operating in State RW, that is when data is being read from MEM 10112 and written into SR 10362 or RCWS 10358. In this case, however, low order 3 bits of destination register address, that is register select field, are provided by RDS 25324, which decodes microinstruction word subfield MD (Memory Destination). RDS 25324 also provides a control input that DSTADR 25318 to select one of Current, Previous, or Bottom pointers from MISPR 10356 to comprise frame select field of destination register address.

As stated above, frame select field of source and destination register addresses are provided from TOSCNT 25310, TOS-1CNT 25312, and BOSCNT 25314. As described above, the most significant bit of source and destination register address are forced to logic 1 or logic 0, depending upon whether GR 10360 or SR 10362, BIAS 20246, and RCWS 10358 are being addressed. Contents of TOSCNT 25310 to BOSCNT 25314, that is Current, Previous, and Bottom Pointers, are controlled by microinstruction control outputs of FUSITT 11012. Current and Previous Pointers change as stacks are "pushed" or "popped" to and from MIS 10368 as JP 10114 performs, respectively, calls and returns. Similarly, Current, Previous and Bottom Pointers will be incremented or decremented as MIS 10368 frames are transferred between MIS 10368 and MEM 10112, as previously described with respect to CS 10110's Stack Mechanisms.

Referring first to Current and Previous Pointer operation, Current and Previous Pointers in TOSCNT 25310 and TOS-1CNT 25312 are initially set, respectively, to point to Frames 1 and 0 of MIS 10368 by being loaded from JPD Bus 10142. TOSCNT 25310 and TOS-1CNT 25312 are enabled to count when two conditions are met. First condition is dependent upon current operating state of CS 10110. TOSCNT 25310 and TOS-1CNT 25312 will be enabled to count during last system clock cycle of CS 10110 operating States M1 or AB. Second condition is dependant upon whether JP 10114 is to execute a call or return. TOSCNT 25310 and TOS-1CNT 25312 may be enabled to count if a current microinstruction indicates JP 10114 is to execute a call or return, or if CS 10110 is exiting State AB as exit from State AB is an implied call operation. Both a call and an implied call, that is exit from State AB, will cause TOSCNT 25310 and TOS-1CNT 25312 to be incremented. A return will cause TOSCNT 25310 and TOS-1CNT 25312 to be decremented.

Referring to BOSCNT 25314, Bottom Frame Pointer is initially loaded from JPD Bus 10142 to point to MIS 10368 Frame 1. Again, incrementing or decrementing of BOSCNT 25314 is dependant upon CS 10110 operating state and operation to be performed. BOSCNT 25314 is enabled to count upon exiting from State M1. In addition, DEVCMD subfield of a current microinstruction word must indicate that BOSCNT 25314 is to be incremented or decremented. BOSCNT 25314 will be incremented or decremented upon exit from

State M1 as indicated by microinstruction word DEVCMD subfield.

SEM 25320 monitors relative values of Current and Bottom Pointers residing in TOSCNT 25310 and BOSCNT 25314 and provides outputs to EVENT 20284 for purposes of controlling operation of MI 10368 and MOS 10370. SEM 25320 is comprised of a Read Only Memory, for example 93S427s, receiving Current and Bottom Pointers as inputs. SEM 25320 detects 3 Events occurring in operation of TOSCNT 25310 and BOSCNT 25314, and thus in operation of MIS 10368 and MOS 10370. First, SEM 25320 detects an MIS 10368 Stack Overflow. This Event is indicated if the present value of Current Frame Pointer is greater than 8 larger than the present value of Bottom Frame Pointer. Second, SEM 25320 detects when MIS 10368 contains only one frame of information. This event is indicated if the value of Current Frame Pointer is equal to the value of Bottom Frame Pointer. In this case, the previous frame of MIS 10368 resides in MEM 10112 and must be fetched from MEM 10112 before a reference to the previous stack frame may be made. Third, SEM 25320 detects when MIS 10368 and MOS 10370 are full. This Event is indicated if the present value of Current Frame Pointer is 16 larger than the present value of Bottom Frame Pointer. When this Event occurs, any further attempt to write a frame onto MIS 10368 or MOS 10370 will result in a MOS 10370 Stack Overflow. EVENT 20284 responds to these Events indicated by SEM 25320 by initiating execution of an appropriate Event Handling microinstruction sequence, as previously described. It should be noted that MIS 10368 and MOS 10370 are addressed in the same manner, that is through use of Current, Previous and Bottom Frame Pointers and certain microinstruction word subfields. Primary difference between operation of MIS 10368 and MOS 10370 is in the manner in which stack overflows are handled. In the case of MIS 10368, stack frames are transferred between MIS 10368 and MEM 10112 so that MIS 10368 is effectively a bottomless stack. MOS 10370, however, contains a maximum of 8 stack frames, in a present embodiment of CS 10110, so that no more than eight Events may be pushed onto MOS 10370 at a given time.

GR 10360 is addressed in a manner similar to SR 10362, BIAS 20246, and RCWS 10358, that is through ADRSRC 25316, DSTADR 25318, and SDADRMUX 25322. Again, register select fields of source and destination register addresses are provided by microinstruction word SRC and DST subfields. Frame select field of source and destination register addresses is, however, specified by microinstruction word CONEXT subfield. In this case, microinstruction word RS and RD subfields specify that frame select fields of source and destination register addresses are to be provided by CONEXT subfield. Accordingly, ADRSRC 25316 and DSTADR 25318 provide CONEXT subfield as SRCFADR and DSTFADR inputs to SDADRMUX 25322.

Having described structure and operation of RAG 20288, TIMERS 20296 will be described next below. Referring to Fig. 254, a partial block diagram of TIMERS 20296 is shown. As indicated therein, TIMERS 20296 includes Interval Timer (INTTMR) 25410, Egg Timer (EGGTMR) 25412, and Egg Timer Clock Enable Gate (EGENB) 25416.

d.d.d. Timers 20296 (Fig. 254)

Referring first to INTTMR 25410, a primary function of INTTMR 25410 is to maintain CS 10110 architectural time as previously described with reference to Fig. 106A and previous descriptions of CS 10110 UID addressing. As described therein, a portion of all UID addresses generated by all CS 10110 systems is an Object Serial Number (OSN) field. OSN field uniquely defines each object created by operation of or for use in a particular CS 10110. OSN field of an object's UID is, in a particular CS 10110, generated by determining time of creation of that object relative to an arbitrary historic starting time common to all CS 10110 systems. That time is maintained within a MEM 10112 storage space, or address location, but is measured by operation of INTTMR 25410.

INTTMR 25410 is a 28 bit counter clocked by a 110 Nano-Second Clock (110NSCLK) input and is enabled to count by a one MHZ Clock Enable input (CLK1MHZENB). INTTMR 25410 may thereby be clocked at a one MHZ rate to measure one microsecond intervals. Maximum time interval which may be measured by INTTMR 25410 is thereby 268.435 seconds.

As indicated in Fig. 254, INTTMR 25410 may be loaded from and read to JPD Bus 10142. In normal operation, the MEM 10112 location containing architectural time for a particular CS 10110 will be loaded with current architectural time at time of start up of that particular CS 10110. INTTMR 25410 will concurrently be loaded with all zeros. Thereafter, INTTMR 25410 will be clocked at one microsecond intervals. Periodically, when INTTMR 25410 overflows, architectural time stored in MEM 10112 will be accordingly updated. At any time, therefore, current architectural time may be determined, down to a one microsecond increment, by reading architectural time from the previous updated architectural time stored in MEM 10112 and elapsed interval since last update of architectural time from INTTMR 25410. In the event of a failure of CS 10110, architectural time in MEM 10112 and INTTMR 25410 may be saved in MEM 10112 by reading elapsed intervals since last architectural time update. When normal CS 10110 operation resumes, INTTMR 25410 may be reloaded with a count reflecting current architectural time. As indicated in Fig. 254, INTTMR 25410 is loaded from JPD Bus 10142 when INTTMR 25410 is enabled by a Load Enable input (LDE) provided from DP 10118.

Referring to EGGTMR 25412, certain CS 10110 Events, in particular Asynchronous Events previously described with reference to EVENT 20284, are received or acknowledged by EVENT 20284 only at conclusion of State M1 of first microinstruction of an SOP. As certain CS 10110 microinstructions have long execution times, these Asynchronous Events may be subjected to an extended latency, or waiting, interval.

before being serviced. EGGTMR 25412, in effect, measures latency time of pending Asynchronous Events and provides an output to EVENT 20284 if a predetermined maximum latency time is exceeded.

As indicated in Fig. 254, EGGTMR 25412 is clocked by a 110 Nano-Second Clock input (110NSCLK). EGGTMR 25412 is initially set to zero by load input (LDZRO) at end of State M1 of the first microinstruction of each SOP executed by CS 10110, or when specifically instructed so by DEVCMD subfield of a microinstruction word. EGGTMR 25412 is incremented when enabled by Clock Enable (CLKENB) input from EGGENB 25416. There are two conditions necessary for EGGTMR 25412 to be incremented. First condition is occurrence of an Asynchronous Event, which is indicated by input ASYEVNT to EGGENB 25416 from EVENT 20284. Second condition is that 16 or more microseconds have elapsed since last increment of EGGTMR 25412. This interval is measured by an output from fourth bit of INTTMR 25410 which, as shown in Fig. 254, is connected to an input of EGGENB 25416. EGGTMR 25412 is a four bit counter and will thereby overflow and generate output OVRFLW to EVENT 20284 256 microseconds after beginning of an SOP if an Asynchronous Event has occurred and if at least 16 microseconds have elapsed since start of that SOP. EGGTMR 25412 thereby insures a maximum service latency of 256 microseconds for Asynchronous Events.

e.e.e. Fetch Unit 10120 Interface to Execute Unit 10122

Finally, as previously described FU 10120's interface to EU 10122 is primarily comprised of EUDIS Bus 20206, for providing EUDPs to EU 10122's EUSITT, and FUINT 20298. Operation of EUSDT 20266 and EUDIS Bus 20206 has been previously described and will be described further in a following description of EU 10122. FUINT 20298 is primarily concerned with generating Event Requests for conditions signalled from EU 10122 so that these Events may be serviced. In this regard, FUINT 20298 is primarily comprised of gates receiving Event Requests from EU 10122 and providing corresponding outputs to EVENT 20284. Another interface function performed by FUINT 20298 is generation of a "transfer complete" signal generated by FU 10122 and provided to EU 10122 to assert that a EU 10122 result read from EU 10122 to FU 10120 has been received. This transfer complete signal indicates to EU 10122 that EU 10122's result register, described in a following description of EU 10122, is available for further use by EU 10122. This transfer complete signal is generated by an output of FUSITT 11012 as part of microinstruction sequences for transferring data from EU 10122 to FU 10120 or MEM 10112.

Having described structure and operation of FU 10120, including DESP 20210, MEMINT 20212, and FUCTL 20214, the structure and operation of EU 10122 will be described next below.

C. Execute Unit 10122 (Figs. 203, 255—268)

As previously described, EU 10122 is an arithmetic processor capable of executing integer, packed and unpacked decimal, and single and double precision floating point arithmetic operations. A primary function of EU 10122 is to relieve FU 10120 of certain arithmetic operations, thus enhancing efficiency of CS 10110.

Transfer of operands from MEM 10112 to EU 10122 is controlled by FU 10120, as is transfer of results of arithmetic operations from EU 10122 to FU 10120 or MEM 10112. In addition, EU 10122 operations are initiated by FU 10120 by EU 10122 Dispatch Pointers invited to EU 10122 by EUSDT 20266. EU 10122 Dispatch Pointers may initiate both arithmetic operations required for execution of SINs and certain EU 10122 operations assisting in handling of CS 10110 events. As previously described, EU 10122 Dispatch Pointers are translated into sequences of microinstructions for controlling EU 10122 by EU 10122's EUSITT which is similar in structure and operation to FUSITT 11012. As will be described further below, EU 10122 includes a command queue for receiving and storing sequences of EU 10122 Dispatch Pointers from FU 10120. In addition, EU 10122 includes a general register file, or scratch pad memory, similar to GRF 10354. EU 10122's general register file is utilized, in part, in EU 10122 Stack Mechanisms similar to FU 10120's SR's 10362.

Referring to Fig. 203, a partial block diagram of EU 10122 is shown. EU 10122's general structure and operation will be described first with reference to Fig. 203. Then EU 10122's structure and operation will be described in further detail with aid of subsequent figures which will be presented as required.

As indicated in Fig. 203, major elements of EU 10122 include Execute Unit Control Logic (EUCL) 20310, Execute Unit IO Buffer (EUIO) 20312, Multiplier Logic (MULT) 20314, Exponent Logic (EXP) 20316, Multiplier Control Logic (MULTCNTL) 20318, and Test and Interface Logic (TSTINT) 20320. EUCL 20310 receives Execute Unit Dispatch Pointers (EUDP's) from EUSDT 20266 and provides corresponding sequences of microinstructions to control operation of EU 10122.

EUIO 20312 receives operands, or data, from MEM 10112, translates those operands into certain formats most efficiently used by EU 10122. EUIO 20312 receives results of EU 10122's operations and translates those results into formats to be returned to MEM 10112 or FU 10120, and presents those results to MEM 10112 and FU 10120.

MULT 20314 and EXP 20316 are arithmetic units for performing arithmetic manipulations of EU 10122 operations. In particular, EXP 20316 performs operations with respect to exponent fields of single and double precision floating point operations. MULT 20314 performs arithmetic manipulations with respect to mantissa fields of single and double precision floating point operations, and arithmetic operations with regard to integer and packed decimal operations. MULTCNTL 20318 controls and coordinates operation of

EP 0 067 556 B1

MULT 20314 and EXP 20316 and prealignment and normalization of mantissa and exponent fields in floating point operations. Finally, TSTINT 20320 performs certain test operations with regard to EU 10122's operations, and is the interface between EU 10122 and FU 10120.

5 a. General Structure of EU 10122

1. Execute Unit I/O 20312

Referring first to EUIO 20312, EUIO 20312 includes Operand Buffer (OPB) 20322, Final Result Output Multiplexer (FROM) 20324, and Exponent Output Multiplexer (EXOM) 20326. OPB 20322 has first and second inputs connected, respectively, from MOD Bus 10144 and JPD Bus 10142. OPB 20322 has a first output connected to a first input of Multiplier Input Multiplexer (MULTIM) 20328 and MULT 20314. A second output of OPB 20322 is connected to first inputs of Inputs Selector A (INSELA) 20330 and Exponent Execute Unit General Register File Input Multiplexer (EXRM) 20332 in EXP 20316.

FROM 20324 has an output connected to JPD Bus 10142. A first input of FROM 20324 is connected from output of Multiplier Execute in General Register File Input Multiplexer (MULTRM) 20334 and MULT 20314. A second input of FROM 20324 is connected from output of Final Result Register (RFR) 20336 of MULT 20314. EXOM 20326 has an output connected to JPD Bus 10142. EXOM 20326 is a first input connected from output of Scale Register (SCALER) 20338 of EXP 20316. EXOM 20326 has second and third inputs connected from outputs of, respectively, Next Address Generator (NAG) 20340 and Command Queue (COMQ) 20342 of EUCL 20310.

2. Execute Unit Control Logic 20310

Referring to EUCL 20310, EUCL 20310 includes NAG 20340, COMQ 20342, Execute Unit S Interpreter Table (EUSITT) 20344, and Microinstruction Control Register and Decode Logic (mCRD) 20346. COMQ 20342 has an input connected from EUDIS Bus 20206 for receiving SDPs from EUSDT 20266. COMQ 20342 has, as described above, a first output connected to a third input of EXOM 20326, and has a second output connected to an input of NAG 20340. NAG 20340 has, as described above, a first output connected to second input of EXOM 20326. NAG 20340 has a second output connected to a first input of EUSITT 20344. As previously described, EUSITT 20344 corresponds to FUSITT 11012 and stores sequences of microinstructions for controlling operation of EU 10122 in response to EU 10122 Dispatch Pointers from FU 10120. EUSITT 20344 has a second input connected from JPD Bus 10142 and has an output connected to input of mCRD 20346. mCRD 20346 includes a register and logic for receiving and decoding microinstructions provided by EUSITT 20344. In addition to an input from EUSITT 20344, mCRD 20346 has first outputs providing decoded microinstruction control signals to all parts of EU 10122. mCRD 20346 also has a second output connected to a first input of Input Selector B (INSELB) 20348 and EXP 20316.

3. Multiplexer Logic 20314

Referring to MULT 20314, MULT 20314 includes two parallel arithmetic operation paths for performing addition, subtraction, multiplication, and division operations on packed decimal numbers, integer numbers, and mantissa portions of single and double precision floating point numbers. MULT 20314 also includes a related portion of EU 10122's general register file, a memory for storing constants used in arithmetic operations, and certain input data selection circuits. That portion of EU 10122's GRF residing in MULT 20314 is comprised of Multiplier Register File (MULTRF) 20350. Output of MULTRF 20350 is connected to a second input of MULTIM 20328. A first input of MULTRF 20350 is connected from output of RFR 20336 and a second input of MULTRF 20350 is connected from output of MULTRM 20334. First and second inputs of MULTRM 20334 are in turn connected, respectively, from output of RFR 20336 and from output of Container Size Logic (CONSIZE) 20352 of TSTINT 20320.

MULTIM 20328 selects the data inputs to MULT 20314's arithmetic circuits and has, as previously described, first and second inputs connected respectively from first output of OPB 20322 and from output of MULTRF 20350. Output of MULTIM 20328 is connected through Multiplier (MULT) Bus 20354 to input of Multiplier Quotient Register (MQR) 20356 and to input of Nibble Shifter (NIBSHF) 20358. Another input to MQR 20356 and NIBSHF 20358 is provided by Constant Store (CONST) 20360. CONST 20360 is a memory for storing constant values used in MULT 20314 operations. Output of CONST 20360 is connected to MULT Bus 20354. MULT 20314's arithmetic circuits may thereby be provided with inputs from OPB 20322, MULTRF 20350, and CONST 20360.

MULT 20314's arithmetic circuitry is comprised of two, parallel arithmetic operation paths having, as common inputs, outputs of MULTIM 20328 and CONST 20360. Common termination of these parallel arithmetic operation paths is Final Register Shifter (FRS) 20362. A first arithmetic operation path is provided through NIBSHF 20358, whose input is connected from MULT Bus 20354. NIBSHF 20358's output is connected to a first input of FRS 20362 and a control input of NRBSHF 20358 is connected from an output of Multiplier Control Logic (MULTCNT) 20364 and MULTCNTL 20318.

MULT 20314's second arithmetic operation path is provided through MQR 20356. As described above, MQR 20356's input is connected from MULT Bus 20354. MQR 20356's output is connected to first and

EP 0 067 556 B1

second inputs of Times 1 And Times 2 Multiply Shifter (MULTSHFT12) 20366 and Times 4 And Times 8 Multiply Shifter (MULTSHFT48) 20368. Outputs of MULTSHFT12 and MULTSHFT8 are connected, respectively, to first and second inputs of First Multiplier Arithmetic and Logic Unit (MULTALU1) 20370. MULTALU1 20370's output is connected to input of Multiplier Working Register (MWR) 20372. Output of MWR 20372 is connected to a first input of Second Multiplier Arithmetic and Logic Unit (MULTALU2) 20374. A second input of MULTALU2 20374 is connected from output of RFR 20336. Output of MULTALU2 is connected to a second input of FRS 20362. As described above, first input of FRS 20362 is connected from output of NIBSHF 20368. Output of FRS 20362 is connected to input of RFR 20336.

As described above, output of RFR 20336 is connected to second input of MULTALU2 20374, to first input of MULTRF 20350, to first input of MULTRM 20334, and to second input of FROM 20324. Output of RFR 20336 is also connected to input of Leading Zero Detector (LZD) 20376 of MULTCNTL 20318, and to inputs of Exception Logic (ECPT) 20378, CONSIZE 20352, and TSTINT 20320.

4. Exponent Logic 20316

Referring to EXP 20316, as previously described EXP 20316 performs certain operations with respect to exponent fields of single and double precision floating point number in EU 10122 floating point operations. EXP 20316 includes a second portion of EU 10122's general register file, shown herein as Exponent Register File (EXPRF) 20380. Although indicated as individual register files, MULTRF 20350 and EXPRF 20380 comprise, as in GRF 10354, a unitary register file structure with common, parallel addressing of corresponding registers therein.

Output of EXPRF 20380 is connected to a second input of INSELA 20330. A first input of EXPRF 20380 is connected from output of EXRM 20332. As previously described, a first input of EXRM 20332 is connected from second output of OPB 20322 through EXPQ Bus 20325. A second input of EXRM 20332 is connected from output Scale Register (SCALER) 20338. A second input of EXPRF 20380 is connected from output of Sign Logic (SIGN) 20382. Input of SIGN 20382 is connected from second output of SCALER 20338.

INSELA 20330, INSELB 20348, Exponent ALU (EXPALU) 20384 and SCALER 20338 comprise EXP 20316's arithmetic circuitry for manipulating exponent fields of floating point numbers. INSELA 20330 and INSELB 20348 select, respectively, first and second inputs to EXPALU 20384. As previously described, a first input of INSELA 20330 is connected from second output of OPB 20322 through EXPQ Bus 20325. Second input of INSELA 20330 is connected from output of EXPRF 20380. Output of INSELA 20330 is connected to first input of EXPALU 20384. First input of INSELB 20348 is, as previously described, connected from a second output of mCRD 20346. Second input of INSELB 20348 is connected from output of OPB 20322 through EXPQ Bus 20325. Third input of INSELB 20348 is connected from output of SCALER 20338 and fourth input of INSELB 20348 is connected from output of LZD 20376. Output of INSELB 20348 is connected to second input of EXPALU 20384. Output of EXPALU 20384 is connected to input of SCALER 20338.

As previously described, second output of SCALER 20338 is connected with input of SIGN 20382 and first output is connected to second input of EXRM 20332 and to third input of INSELB 20348. First output of SCALER 20338 is also connected to EXPQ Bus 20325, to first input of EXOM 20326, and to a second input of MULTCNT 20364.

5. Multiplier Control 20318

As previously described, MULTCNTL 20318 provides certain control signals and information for controlling and coordinating operation of EXP 20316 and MULT 20314 in performing arithmetic operations on floating point numbers. MULTCNTL 20318 includes LZD 20376 and MULTCNT 20364. Input of LZD 20376 is connected from output of RFR 20336 through FR Bus 20337. Output of LZD 20376 are connected to a second input of MULTCNT 20364 and to fourth input of INSELB 20348. A second input of MULTCNT 20364 is connected from output of SCALER 20338. As previously described, control output of MULTCNT 20364 is connected to control inputs of NIBSHF 20358.

6. Test and Interface Logic 20320

Finally, TSTINT 20320 includes ECPT 20378, CONSIZE 20352, and Testing Condition Logic (TSTCON) 20386. Input of ECPT 20378 and first input of CONSIZE 20352 are connected from output of RFR 20336 through FR Bus 20337. A second input of CONSIZE 20352 is connected from LENGTH Bus 20226. An output of CONSIZE 20352 is connected, together with other inputs from EU 10122 (not shown for clarity of presentation) to TSTCON 20386. Output of TSTCON 20386 (not shown for clarity of presentation) are connected to NAG 20340. TSTCON 20386 and ECPT 20378 have outputs to and inputs from FU 10120's FUINT 20298.

Having described the overall structure of EU 10122 above, operation of EU 10122 will be described next below with aid of further diagrams which will be introduced as required. Finally, operation of TSTINT 20320 will be described, including a description of the detailed control signal interface between EU 10122 and FU 10120 through TSTINT 20320 and FUINT 20298. In addition to defining the interface between EU 10122 and FU 10120, certain features of EU 10122 operation will be described wherein those operations are executed

EP 0 067 556 B1

in cooperation with MEM 10112 and FU 10120. For example, EU 10122's Stack Mechanisms, comprising in part portions of MULTRF 20350 and EXPRF 20380, resides partly in MEM 10112 so that operation of EU 10122's Stack Mechanisms requires cooperative operations by EU 10122, MEM 10112 and FU 10120.

5

b. Execute Unit 10122 Operation (Fig. 255)

1. Execute Unit Control Logic 20310 (Fig. 255)

Referring to Fig. 255, a more detailed block diagram of EUCL 20310 is shown. As described above, EUCL 20310 receives EU 10122 Dispatch Pointers through EUDIS Bus 20206 from EUSDT 20266 and FUCTL 20214. EU 10122 Dispatch Pointers select certain EU 10122 microinstruction sequences for executing EU 10122 arithmetic operations as required to execute user's programs, that is SOPs, and to assist in handling JP 10114 Events. As described above, major elements of EUCL 20310 include COMQ 20342, EUSITT 20344, mCRD 20346, and NAG 20340.

15

a.a. Command Queue 20342

Inputs of COMQ 20342 are connected from EUDIS Bus 20206 to receive and store EU 10122 Dispatch pointers provided from EUSDT 20266. Each such EU 10122 Dispatch Pointer is comprised of two information fields. A first information field contains a 10 bit starting address of a corresponding sequence of microinstructions residing in EUSITT 20344. Second field of each EU 10122 Dispatch Pointer is a 6 bit field containing certain control information, such as information identifying data format of corresponding operands to be operated upon. In this case unit dispatch pointer control field bits specify whether operands to be operated upon comprise signed or unsigned integer, packed or unpacked decimal, or single or double precision floating point numbers.

COMQ 20342 is comprised of two one word wide by two word deep register files. A first of these register fields is comprised of SOP Command Queue Control Store (CQCS) 25510 and SOP Command Queue Address Store (CQAS) 25512. Together, CQCS 25510 and CQAS 25512 comprise a one word wide by two word deep register file for receiving and storing EU 10122 Dispatch Pointers corresponding to SOPs, that is Dispatch Pointers for initiating EU 10122 operations directly concerned with executing a user's program. Address fields of these SOPs are received in CQAS 25512, while control fields are received and stored in CQCS 25510. COMQ 20342 is thereby capable of receiving and storing up to two sequential EU 10122 Dispatch Pointers corresponding to user program SOPs. These SOP derived Dispatch Pointers are executed in the order received from FU 10120. EU 10122 is thereby capable of receiving and storing one currently executing SOP Dispatch Pointer and one pending SOP Dispatch Pointer. Further SOP Dispatch Pointers may be read into COMQ 20342 as previous SOPs are executed.

35

b.b. Command Queue Event Control Store 25514 and Command Queue Event Address Control Store 25516

Command Queue Event Control Store (CQCE) 25514 and command Queue Event Address Control Store (CQAE) 25516 are similar in function and operation to, respectively, CQCS 25510 and CQAS 25512. CQCE 25514 and CQAE 25516 receive and store, however, EU 10122 Dispatch Pointers initiating EU 10122 operations requested by FU 10120 as required to handle JP 10114 Events. Again, CQCE 25514 and CQAE 25516 comprise a one word wide by two word deep register file. CQAE 25516 receives and stores address fields of Event Dispatch Pointers, while CQCE 25514 receives and stores corresponding control fields of Event Dispatch Pointers. Again, COMQ 20342 is capable of receiving and storing up to two sequential Event Dispatch Pointers at a time.

As indicated in Fig. 255, outputs of CQAS 25512 and CQAE 25516, that is address fields of EU 10122 Dispatch Pointers are provided as inputs to Select Case Multiplexer (SCASE) 25518 and Starting Address Select Multiplexer (SAS) 25520 and NAG 20340, which will be described further below. Control field outputs of CQCS 25510 and CQCE 25514 are provided as inputs to OPB 20322, described further below.

50

c.c. Execute Unit S-Interpreter Table 20344

Referring to EUSITT 20344, as described above EUSITT 20344 is a memory for storing sequences of microinstructions for controlling operation of EU 10122 in response to EU 10122 Dispatch Pointers received from FU 10120. These microinstruction sequences may, in general, direct operation of EU 10122 to execute arithmetic operations in response to SOPs of user's programs, or aid direct execution of EU 10122 operations required to service JP 10114 Events. EUSITT 20344 may be, for example, a 60 bit wide by 1,280 word long memory structured as pages of 128 words per page. A portion of EUSITT 20344's pages may be contained in Read Only Memory, for example for storing sequence of microinstructions for handling JP 10114 Events. Remaining portions of EUSITT 20344 may be constructed of Random Access Memory, for example for storing sequences of microinstructions for executing EU 10122 operations in response to user program SOPs. This structure allows EU 10122 microinstruction sequences concerned with operation of JP 10114's internal mechanisms, for example handling of JP 10114 Events, to be effectively permanently

65

EP 0 067 556 B1

stored in EUSITT 20344. That portion of EUSITT 20344 constructed of Random Access Memory may be used to store sequences of microinstructions for executing SOPs. These Random Access Memories may be used as writable control store to allow sequences of microinstructions for executing SOPs of one or more S-Languages currently being utilized by CS 10110 to be written into EUSITT 20344 from MEM 10112 as required.

As previously described, EUSITT 20344's second input is a Data (DATA) input connected from JPD Bus 10142. EUSITT 20344's data input is utilized to write sequences of microinstructions into EUSITT 20344 from MEM 10112 through JPD Bus 10142. EUSITT 20344's first input is an address (ADR) input connected from output of Address Driver (ADR) 25522 and NAG 20340. Address inputs provided by ADR 25522 select word locations within EUSITT 20344 for writing of microinstructions into EUSITT 20344, or for reading of microinstructions from EUSITT 20344 to mCRD 20346 to control operation of EU 10122. Generation of these address inputs to EUSITT 20344 by NAG 20340 will be described further below.

d.d. Microcode Control Decode Register 20346

Output of EUSITT 20344 is connected to input of mCRD 20346. As previously described, mCRD 20346 is a register for receiving microinstructions from EUSITT 20344, and decoding logic for decoding those microinstructions and providing corresponding control signals to EU 10122. As indicated in Fig. 255, Diagnostic processor Micro-Program Register (DPmR) 25524 is a 60 bit register connected in parallel with output of EUSITT 20344 to input of mCRD 20346. DPmR 25524 may be loaded with 60 bit microinstructions by DP 10118. Diagnostic microinstructions may thereby be provided directly to input of mCRD 20346 to provide direct microinstruction by microinstruction control of EU 10122.

Outputs of mCRD 20346 are provided, in general, to all portions of EU 10122 to control detailed operations of EU 10122. Certain outputs of mCRD 20346 are connected to inputs of Next Address Source Select Multiplexer (NASS) 25526 and Long Branch Page Address Gate (LBPAG) 25528 and NAG 20340. As will be described further below, these outputs of mCRD 20346 are used in generating address inputs to EUSITT 20344 when particular microinstructions sequences call for Jumps or Long Branches to other microinstruction sequences. Outputs of mCRD 20346 are also connected in parallel to inputs of Execution Unit Micro-Instruction Parity Check Logic (EUmIPC) 25530. EUmIPC 25530 checks parity of all microinstruction outputs of mCRD 20346 to detected errors in mCRD 20346's outputs.

e.e. Next Address Generator 20340

As described above, read and write addresses to EUSITT 20344 provided by NAG 20340 through ADRD 25522. Address inputs to ADRD 25522 are provided from either NASS 25526 or Diagnostic Processor Address Register (DPAR) 25532. In normal operation, address inputs to EUSITT 20344 are provided from NASS 25526 as will be described momentarily. DP 10118, however, may load EUSITT 20344 addresses into DPAR 25532. These addresses may then be read from DPAR 25532 through ADRD 25522 to individually select address locations within EUSITT 20344. DPAR 25532 may be utilized, in particular, to provide addresses to allow stepping through of EU 10122 microinstruction sequences microinstruction by microinstruction.

As described above, NASS 25526 is a multiplexer having inputs from three NAG 20340 address sources. NASS 25526's first address input is from Jump (JMP) output of mCRD 20346 and LBPAG 25528. These address inputs are utilized, in part, when a current microinstruction calls for a Jump or Long Branch to another microinstruction or microinstruction sequence. Second address source is provided from SAS 25520 and, in general, is comprised of starting addresses of microinstruction sequences. SAS 25520 is a multiplexer having a first input from COAS 25512 and COAE 25516, that is starting addresses of microinstruction sequences corresponding to SOPs or for servicing JP 10114 Events. A second SAS 25520 input is provided from Sub-routine Return Address Stack (SUBRA) 25534. In general, and as will be described further below, SUBRA 25534 operates as a stack mechanism for storing current microinstruction addresses of interrupted microinstruction sequences. These stored addresses may subsequently be utilized to resume execution of those interrupted microinstruction sequences. Third address source to NASS 25526 is provided from Sequential and Case Address Generator (SCAG) 25536. In general, SCAG 25536 generates address to select sequential microinstructions within particular microinstruction sequences. SCAG 25536 also generates microinstruction address for microinstruction Case operations. As indicated in Fig. 255, outputs of SCAG 25536 and of SAS 25520 are bused together to comprise a single NASS 25526 input. Selection between outputs of SCAG 25536 and SAS 25520 are provided by control inputs (not shown for clarity of presentation) to SCAG 25536 and SAS 25520. Selection between NASS 25526's address inputs is controlled by Next Address Source Select Control Logic (NASSC) 25538, which provides control inputs to NASS 25526. NASSC 25538 is effectively a multiplexer receiving control inputs from TSTCON 20386 and TSTINT 20320. As will be described further below, TSTCON 20386 monitors certain operating conditions or states within EU 10122 and provides corresponding inputs to NASSC 25538. NASSC 25538 effectively decodes these control inputs from TSTCON 20386 to provide selection control input to NASS 25526.

Having described overall structure and operation of NAG 20340, operation of NAG 20340 will be

described in further detail next below.

Referring first to NASS 25526's address inputs provided from JMP output of mCRD 20346 and LBPAG 25528, this address source is provided to allow selection of a next microinstruction by a current microinstruction. JMP output of mCRD 20346 allows a current microinstruction to direct a Jump to another microinstruction within the same page of EUSITT 20344. NASS 25526's input through LBPAG 25528 is provided from another portion of mCRD 20346's output specifying pages within EUSITT 20344. This input through LBPAG 25528 allows execution of Long Branch operations, that is jumps from a microinstruction in one page of EUSITT 20344 to a microinstruction in another page. In addition, NASS 25526's input from JMP output of mCRD 20346 and through LBPAG 25528 is utilized to execute an Idle, or Standby, routine when EU 10122 is not currently executing a microinstruction sequence requested by FU 10120. In this case, Idle routine directs TSTCON 20386 to monitor EU 10122 Dispatch Pointer inputs to EU 10122 from FU 10120. If no EU 10122 Dispatch Pointers are present in COMQ 20342, or none are pending, TSTCON 20386 will direct NASSC 25538 to provide control inputs to NASS 25526 to select NASS 25526's input from mCRD 20346 and LBPAG 25528. Idle routine will continually test for EU 10122 Dispatch pointer inputs until such a Dispatch Pointer is received into COMQ 20342. At this time, TSTCON 20386 will detect the pending Dispatch Pointer and direct NASS 25538 to provide control outputs to NASS 25526 to select NASS 25526's input from, in general, SAS 25520. TSTCOND 20386 and NASSC 25538 will also direct NASS 25526 to select inputs from SAS 25520 upon return from a called microinstruction to a previously interrupted microinstruction sequence.

As described above, SAS 25520 receives starting addresses from COMQ 20342 and from SUBRA 25534. SAS 25520 will select the output of CQAS 25512 or of CQAE 25516 as the input to NASS 25526 when a new microinstruction sequence is to be initiated to execute a user's program SOP or to service a JP 10114 Event. SAS 25520 will select an address output of SUBRA 25534 upon return from a called sub-routine to a previously executing but interrupted sub-routine. SUBRA 25534, as described above, is effectively a stack mechanism for storing addresses of currently executing microinstructions when those microinstruction sequences are interrupted. SUBRA 25534 is an 11 bit wide by 8 word deep register with certain registers dedicated for use in stacking Event Handling microinstruction sequences. Other portions of SUBRA 25534 are utilized for stacking of microinstruction sequences for executing SOPs, that is for stacking microinstruction sequences wherein a first microinstruction sequence calls for a second microinstruction sequence. SUBRA 25534 is not operated as a first-in-first out stack, but as a random access memory wherein address inputs selecting registers and SUBRA 25534 are provided by microinstruction control outputs of mCRD 20346. Operations of SUBRA 25534 as a stack mechanism is thereby controlled by the microinstruction sequences stored in EUSITT 20344. As indicated in Fig. 255, addresses of current microinstructions of interrupted microinstruction sequences are provided to data input of SUBRA 25534 from output of SCAG 25536, which will be described next below.

As described above, SCAG 25536 generates sequential addresses to select sequential microinstructions within microinstruction sequences and to generate microinstruction addresses for Case operations. SCAG 25536 includes Next Address Register (NXTR) 25540, Next Address Arithmetic and Logic Unit (NAALU) 25542, and SCASE 25518. NAALU 25542 is a 12 bit arithmetic and logic unit. A first eleven bit input of NAALU 25542 is connected from output of ADRD 25522 and is thereby current address provided to EUSITT 20344. A second four bit input to NAALU 25542 is provided from output of SCASE 25518. During sequential execution of a microinstruction sequence, output of SCASE 25518 is binary zeros and carry input of NAALU is forced to 1. Output of NAALU 25542 will thereby be and address one greater than the current microinstruction address provided to EUSITT 20344 and will thereby be the address of the next sequential microinstruction. As indicated in Fig. 255, SCASE 25518 receives an input from output of SCALER 20338. This input is utilized during Case operations and allows a data sensitive number to be selected as SCASE 25518's output into second input of NAALU 25542. SCASE 25518's input from SCALER 20338 thereby allows NAG 20340 to perform microinstruction Case operations wherein Case Values are determined by the contents of SCALER 20338.

Next address outputs of NAALU 25542 are loaded into NXTR 25540, which is comprised of tri-state output registers. Next address outputs of NXTR 25540 are connected, in common with outputs of SAS 25520, to second input of NASS 25526 as described above. During normal execution of microinstruction sequences, therefore, SCAG 25536 will, through NASS 25526 and ADRD 25522, select sequential microinstructions from EUSITT 20344. SCAG 25536 may also, as just described, provide next microinstruction addresses in microinstruction Case operations.

In summary, NAG 20340 is capable of performing all usual microinstruction sequence addressing operations. For example, NAG 20340 allows selection of next microinstructions by current microinstructions, either for Jump operations or Long Branch operations, through NASS 25526's input from mCRD 20346's JMP or through LBPAG 25528. NAG 20340 may provide microinstruction sequence starting addresses through COMQ 20342 and SAS 25520, or may provide return addresses to interrupted and stacked microinstruction sequences through SUBRA 25534 and SAS 25520. NAG 20340 may sequentially address microinstructions of a particular microinstruction sequence through operation of SCAG 25536, or may perform microinstruction Case operations through SCAG 25536.

2. Operand Buffer 20322

Having described structure and operation of EUCL 20310, structure and operation of OPB 20322 will be described next below. As previously described, OPB 20322 receives operands, that is data, from MEM 10112 and FU 10120 through MOD Bus 10144 and JPD Bus 10142. OPB 20322 may then perform certain operand format translations to provide data to MULT 20314 and EXP 20316 in the formats most efficiently utilized by MULT 20314 and EXP 20316. As previously described, EU 10122 may perform arithmetic operations on integer, packed and unpacked decimal, and single or double precision floating point numbers.

In summary, therefore, OPB 20322 is capable of accepting integer, single and double precision floating point, and packed and unpacked decimal operands from MEM 10112 and FU 10120 and providing appropriate fields of those operands to MULT 20314 and EXP 20316 in the formats most efficiently utilized by MULT 20314 and EXP 20316. In doing so, OPB 20322 extracts exponent and mantissa fields from single and double precision floating point operands to provide exponent and mantissa fields of these operands to, respectively, EXP 20316 and MULT 20314, and also unpacks, or converts, unpacked decimal operands to packed decimal operands most efficiently utilized by MULT 20314.

Having described structure and operation of OPB 20322, structure and operation of MULT 20314 will be described next below.

3. Multiplier 20314 (Figs. 257, 258)

MULT 20314, as previously described, performs addition, subtraction, multiplication, and division operations on mantissa fields of single and double precision floating point operands, integer operands, and decimal operands. As described above with reference to OPB 20322, OPB 20322 converts unpacked decimal operands to packed decimal operands to be operated upon by MULT 20314. MULT 20314 is thereby effectively capable of performing all arithmetic operations on unpacked decimal operands.

a.a. Multiplier 20314 Data Paths and Memory (Fig. 257)

Referring to Fig. 257, a more detailed block diagram of MULT 20314's data paths and memory is shown. As previously described, major elements of MULT 20314 include memory elements comprised of MULTRF 20350 and CONST 20360, operand input and result output multiplexing logic including MULTIM 20328 and MULTRM 20334, and arithmetic operation logic. MULT 20314's operand input and result output multiplexing logic and memory elements will be described first, followed by description of MULT 20314's arithmetic operation logic.

As previously described, input data, including operands, is provided to MULT 20314's arithmetic operation logic through MULTIN Bus 20354. MULTIN Bus 20354 may be provided with data from three sources. A first source is CONST 20360 which is a 512 word by 32 bit wide Read Only Memory. CONST 20360 is utilized to store constants used in arithmetic operations. In particular, CONST 20360 stores zone fields for unpacked decimal, that is ASCII character, operands. As previously described, unpacked decimal operands are received by OPB 20322 and converted to packed decimal operands for more efficient utilization by MULT 20314. As such, final result outputs generated by MULT 20314 from such operands are in packed decimal format. As will be described below, MULT 20314 may be utilized to convert these packed decimal results into unpacked decimal results by insertion of zone fields. As indicated in Fig. 257, address inputs are provided to CONST 20360 from EXPQ Bus 20325 and from output of mCRD 20346. Selection between these address inputs is provided through CONST Address Multiplexer (CONSTAM) 25710. CONST 20360 addresses will, in general, be provided from EUCL 20310 but alternately may be provided from EXPQ Bus 20325 for special operations.

Operand data is provided to MULTIN Bus 20354 through MULTIM 20328, which is a dual input, 64 bit multiplexer. A first input of MULTIM 20328 is provided from OPQ Bus 20323 and is comprised of operand information provided from OPB 20322. OPQ Bus 20323 is a 56 bit wide bus and operand data appearing thereon may be comprised of 32 bit integer operands; 32 bit packed decimal operands, either provided directly from OPB 20322 or as a result of OPB 20322's conversion of an unpacked decimal to a packed decimal operand; 24 bit single precision operand mantissa fields; or 56 bit double precision floating point operand mantissa fields. As previously described, certain OPQ Bus 20323 may be zero or sign extension filled, depending upon the particular operand.

Second input of MULTIM 20328 is provided from MULTRF 20350. MULTRF 20350 is a 16 word by 64 bit wide random access memory. As indicated in Figs. 203 and 257, MULTRF 20350 is connected between output of RFR 20336, through FR Bus 20337, and to input of MULT 20314's arithmetic operation logic through MULTIM 20328 and MULTIN Bus 20354. MULTRF 20350 may therefore be utilized as a scratch pad memory for storing intermediate results of arithmetic operations, including reiterative arithmetic operations. In addition, a portion of MULTRF 20350 is utilized, as in GRF 10354, as an EU 10122 Stack Mechanism similar to MIS 10368 and MOS 10370 in FU 10120. Operation of EU 10122 Stack Mechanism will be described in a following description of EU 10122's interfaces to MEM 10112 and FU 10120. Address Inputs (ADR) of MULTRF 20350 are provided from Multiplier Register File Address Multiplexer (MULTRFAM) 25712.

MULTRFAM 25712 is a dual four bit multiplexer comprised, for example, of SN74S258s. In addition to address inputs to MULTRF 20350, MULTRFAM 25712 provides address inputs to EXPRF 20380. As previously described, MULTRF 20350 and EXPRF 20380 together comprise an EU 10122 general register file similar to GRF 10354 and FU 10120. As such, MULTRF 20350 and EXPRF 20380 are addressed in parallel to read and write parallel entries from and to MULTRF 20350 and EXPRF 20380. Address inputs to MULTRFAM 25712 are provided, first, from outputs of mCRD 20346, thus providing microinstruction control of addressing of MULTRF 20350 and EXPRF 20380. Second address input to MULTRFAM 25712 is provided from output of Multiplier Register File Address Counter (MULTRFAC) 25714.

MULTRFAC 25714 is a four bit counter and is used to generate sequential addresses to MULTRF 20350 and EXPRF 20380. Initial addresses are loaded into MULTRFAC 25714 from Multiplier Register File Address Counter Multiplexer (MULTRFACM) 25716. MULTRFACM 25716 is a dual four bit multiplexer. Inputs to MULTRFACM 25716 are provided, first, from outputs of mCRD 20346. This input allows microinstruction selection of an initial address to be loaded into MULTRFAC 25714 to be subsequently used and generating sequential MULTRF 20350 and EXPRF 20380 addresses. Second address input to MULTRFACM 25716 is provided from OPQ Bus 20323. MULTRFACM 25716's input from OPQ Bus 20323 allows a single address, or a starting address of a sequence of addresses, to be selected through JPD Bus 10142 or MOD Bus 10144, for example from MEM 10112 or FU 10120.

Intermediate and final result outputs of MULT 20314 arithmetic logic are provided to data inputs of MULTRF 20350 directly from FR Bus 20337 and from MULTRM 20334. Inputs to MULTRM 20334, in turn, are provided from FR Bus 20337 and from output of CONSIZE 20352 and TSTINT 20320.

FR Bus 20337 is a 64 bit bus connected from 64 bit output of RFR 20336 and carries final and intermediate results of MULT 20314 arithmetic operations. As will become apparent in a following description of MULT 20314 arithmetic operation logic, RFR 20336 output, and thus FR Bus 20337, are 64 bits wide. Sixty-four bits are provided to insure retention of all significant data bits of certain MULT 20314 arithmetic operation intermediate results, in particular operations involving double precision floating point 64 bit mantissa fields. In addition, as will be described momentarily and has been previously stated, MULT 20314 may convert a final result in packed decimal format into a final result in unpacked decimal format. In this operation, a single 32 bit, or one word, packed decimal result is converted into a 64 bit, or two word, unpacked decimal format by insertion of zone fields.

As described above, two parallel data paths are provided to transfer information from FR Bus 20337 into MULTRF 20350. First path is directly from FR Bus 20337 and second path is through Unpacked Decimal Multiplexer (UPDM) 25718 of MULTRM 20334. Direct path is utilized for thirty-two bits of information comprising bits 0 to 23 and bits 56 to 63 of FR Bus 20337. Data path through UPDM 25718 may comprise either bits 24 to 55 of FR Bus 20337, which are connected into a first input of UPDM 25718, or bits 40 through 55 which are connected to a second input of UPDM 25718. Single precision floating point numbers are 32 bit numbers plus two or more guard bits and are thus written into MULTRF 20350 through bits 0 to 23 of the direct path into MULTRF 20350 and through first input (bits 24 to 55) of UPDM 25718. Double precision floating point numbers are 5 bits wide, plus guard bits, and thus utilize the direct path into MULTRF 20350 and the path through first input of UPDM 25718. Bits 56 to 63 of direct path are utilized for guard bits of double precision floating point numbers. Both integer and packed decimal numbers utilize bits 24 through 55 of FR Bus 20337, and are thus written into MULTRF 20350 through first input of UPDM 25718. As previously described, bits 0 to 23 of these operands are filled by sign extension.

a.a.a. Container Size Check

As stated above, MULTRM 20334 has an input from CONSIZE 20352. As will be described below with reference to TSTINT 20320, CONSIZE 20352 performs a "container size" check upon each store back of results from EU 10122 to MEM 10112. CONSIZE 20352 compares the number of significant bits in a result to be stored back to the logical descriptor describing the MEM 10112 address space that result is to be written into. Where reiterative write operations to MEM 10112 are required to transfer a result into MEM 10112, that is a string transfer, container size information may read from CONSIZE 20352 through Container Size Driver (CONSIDED) 25720 and MULTRM 20334 and written into MULTRF 20350. This allows EU 10122, using container size information stored in MULTRM 20334, to perform continuous container size checking during a string transfer of result from EU 10122 to MEM 10112. In addition, as will be described momentarily, container size information may be read from CONSIZE 20352 to JPD Bus 10144.

b.b.b. Final Result Output Multiplexer 20324

Referring finally to FROM 20324, as previously described FROM 20324 is utilized to transfer, in general, results of EU 10122 arithmetic operations onto JPD Bus 10142 for transfer to MEM 10112 or FU 10120. As indicated in Fig. 257, FROM 20324 is comprised of 24 bit Final Result Bus Driver (FRBD) 25722 and Result Bus Driver (RBR) 25724. Input of FRBD 25722 is connected from FR Bus 20337 and allows data appearing thereon to be transferred onto JPD Bus 10142. In particular, FRBD 25722 is utilized to transfer 24 bit mantissa fields of single precision floating point results onto JPD Bus 10142 in parallel with a corresponding exponent field from EXP 20316. RBR 25724 input is connected from RSLT Bus 20388 to allow

output of UPDM 25718 to be transferred onto JPD Bus 10142. RBR 25724, RSLT Bus 20388, and UPDM 25718 are used, in general, to transfer final results of EU 10122 operations from output of MULT 20314 onto JPD Bus 10142. Final results transferred by this data path include integer, packed and unpacked decimal results, and mantissa fields of double precision floating point results. Both unpacked decimal numbers and mantissa fields of double precision floating point numbers are comprised of two 32 bit words and are thus transferred onto JPD Bus 10142 in two sequential transfer operations.

Having described structure and operation of MULT 20314's memory elements and input and output circuitry, MULT 20314's arithmetic operation logic will be described next below.

4. Test and Interface Logic 20320 (Figs. 260—268)

As previously described, TSTINT 20320 includes CONSIZE 20352, ECPT 20328, TSTCOND 20384, and INTRPT 20388. CONSIZE 20352, as previously described, performs "container size" check operations when results of EU 10122 operations are to be written into MEM 10112. That is, CONSIZE 20352 compares size or number of significant bits, of an EU 10122 result to the capacity, or container size, of the MEM 10112 location that EU 10122 result is to be written into. As indicated, in Fig. 203, CONSIZE 20352 receives a first input, that is the results of EU 10122 operations, from FR Bus 20337. A second input of CONSIZE 20352 is connected to LENGTH Bus 20226 to receive length field of logical descriptors identifying MEM 10112 address space into which those EU 10122 results are to be written. CONSIZE 20352 includes logic circuitry, for example a combination of Read Only Memory and Field Programmable Logic Arrays, for examining EU 10122 operation results appearing on FR Bus 20337 and determining the number of bits of data in those results. CONSIZE 20352 compares EU 10122 result size to logical descriptor length field and, in particular, if result size exceeds logical descriptor length, provides an alarm output to ECPT 20328, described below.

TSTCOND 20384, previously described and which will be described further below, is an interface circuit between FU 10120 and EU 10122. TSTCOND 20384 allows FU 10120 to specify and examine results of certain test operations performed by EU 10122 with respect to EU 10122 operations.

ECPT 20328 monitors certain EU 10122 operations and provides outputs indicating when certain "exceptions" have occurred. These exceptions include attempted divisions by zero, floating point exponent underflow or overflow, and integer container size fault.

INTRPT 20388 is again an interface between EU 10122 and FU 10120 allowing FU 10120 to interrupt EU 10122 operations. INTRPT 20388 allows FU 10120 to direct EU 10122 to execute certain operations to aid in handling of certain FU 10120 events previously described.

Operation of CONSIZE 20352, ECPT 20328, TSTCOND 20384, INTRPT 20388, and other features of EU 10122's interface with FU 10120 will be described further below in the following description of operation of that interface and of operation of certain EU 10122 internal mechanisms, such as FU 10120 Stack Mechanisms.

a.a. FU 10120/EU 10122 Interface

As previously described, EU 10122 and FU 10120 are asynchronous processors, each operating under its own microcode control. EU 10122 and FU 10120 operate simultaneously and independently of each other but are coupled, and their operations coordinated, by interface signals described below. Should EU 10122 not be able to respond immediately to a request from FU 10120, FU 10120 will idle until EU 10122 becomes available; conversely, should EU 10122 not receive, or have present, operands or a request for operations from FU 10120, EU 10122 will remain in idle state until operands and requests for operations are received from FU 10120.

In normal operation, EU 10122 manipulates operands under control of FU 10120, which in turn is under control of SOPs of a user's program. When FU 10120 requires arithmetic or logical manipulation of an operand, FU 10120 dispatches a command, that is an Execute Unit Dispatch Pointer (EUDP) to EU 10122. As previously described, an EUDP is basically an initial address into EUSITT 20344. An EUDP identifies starting location of a EU 10122 microinstruction sequence performing the required operation upon operands. Operands are fetched from MEM 10112 under FU 10120 control, as previously described, and are transferred into OPB 20322. Those operands are then called from OPB 20322 by EU 10122 and transferred into MULT 20314 and EXP 20316 as previously described. After the required operation is completed, FU 10120 is notified that a result is ready. At this point, FU 10120 may check certain test conditions, for example through TSTCOND 20384, such as whether an integer or decimal carry bit is set or whether a mantissa sign bit is set or reset. This test operation is utilized by FU 10120 for conditional branching and synchronization of FU 10120 and EU 10122 operations. Exception checking, by ECPT 20328, is also performed at this time. Exception checking determines, for example, whether division by zero was attempted or if a container size fault has occurred. In general, FU 10120 is not informed of exception errors until FU 10120 requests exception checking. After results are transferred into FU 10120 or MEM 10112 by EU 10122, EU 10122 goes to idle operation until a next operation is requested by FU 10120.

Having briefly described overall interface operation between FU 10120 and EU 10122, operation of that interface, referred to as handshaking, will be described in greater detail next below. In general, handshaking operation between EU 10122 and FU 10120 during normal operation may be regarded as following into six operations. These operations may include, for example, loading of COMQ 20342, loading of OPB 20322, storeback or transfer of results from EU 10122 to FU 10120 or MEM 10112, check of test

EP 0 067 556 B1

conditions, exception checking, and EU 10122 idle operation. Handshaking between FU 10120 and EU 10122 will be described below for each of these classes of operation, in the order just referred to.

5 a.a.a. Loading of Command Queue 20342 (Fig. 260)

Referring to Fig. 260, a schematic representation of EU 10122's interface with FU 10120 for purposes of loading COMQ 20342 as shown. During normal SOP directed JP 10114 operation, 8 bit operation (OP) codes are parsed from the instruction stream, as previously described, and concatenated with dialect information to address EUSDT 20266 also as previously described. EUSDT 20266 provides corresponding addresses, that is EUDPs, to EUSITT 20344.

10 Dialect information specifies the S-Language currently being executed and, consequently, the group of microinstruction sequences available in EUSITT 20344 for that S-Language. As previously described, FU 10120 may specify four S-Language dialects with up to 256 EU 10122 microinstruction sequences per dialect, or 8 dialects with up to 128 microinstruction sequences per dialect.

15 EUDPs provided by EUSDT 20266 are comprised of a 9 bit address field, a 2 bit operand information field, and a 1 bit flag field, as previously described. Address field is starting address of a microinstruction sequence in EUSITT 20344 and EU 10122 will perform the operation directed by that microinstruction sequence. EUSITT 20344 requires 11 bits of address field and the 9 bit address field of EUDPs are mapped into an 11 bit address field by left justification and zero filling.

20 FU 10120 may also dispatch, or select, any EU 10122 microinstruction controlled operation from JPD Bus 10142. Such EUDPs are provided from JPD Bus 10142 to data input of EUSITT 20344 and passed directly through to mCRD 20346. Before a EUDP may be provided from JPD Bus 10142, however, FU 10120 provides a check operation comparing that EUDP to a list of legal, or allowed, EUSITT 20344 addresses stored in MEM 10112. A fault will be indicated if an EUDP provided through JPD Bus 10142 is not a legal EUSITT 20344 address. Alternately, FU 10120 may effectively provide an EUDP, or EUSITT 20344 addresses, from a literal field in a FU 10120 microinstruction word. Such a FU 10120 microinstruction word literal field may be effectively utilized as an SOP into EUSDT 20266.

25 Handshaking between EU 10122 and FU 10120 during load COMQ 20342 operations may proceed as illustrated in Fig. 260. A twelve bit EUDP may be placed on EUDIS Bus 20206 and Control Signal Load Command Queue (LDCMQ) asserted. If COMQ 20342 is full, EU 10122 raises control signal Command Hold (CMDHOLD) which causes FU 10120 to remain in State M0 until there is room in COMQ 20342. As previously described, COMQ 20342 is comprised of two, two word buffers wherein one buffer is utilized for normal SOP operation and the other utilized for control of FU 10120 and EU 10122 internal mechanism operation.

30 EUDPs are loaded into COMQ 20342 when state timing signals M1CPT and M1 are asserted. If a EUDP being transferred into COMQ 20342 concerns a double precision floating point operation, control signal Set Double Precision (SETDP) is asserted. SETDP is utilized to control OPB 20322, and because single precision and precision floating point operations otherwise utilize the same SOP and thus would otherwise refer to same EUSITT 20344 microinstruction sequence.

35 At this point, a EUDP has been loaded into COMQ 20342 and will be decoded to control FU 10120 operation by EUCL 20310 as previously described. Each particular EUDP will be cleared by that EUDPs EUSITT 20344 microinstruction sequence after the requested microinstruction sequence has been executed.

45 b.b.b. Loading of Operand Buffer 20320 (Fig. 261)

Referring to Fig. 261, a diagramic representation of the interface and handshaking between EU 10122, FU 10120 and MEM 10112 for loading OPB 20322 is shown. Control signal Clear Queue Full (CLQF) from EU 10122 must be asserted by EU 10122 before FU 10120 initiates a request to MEM 10112 for an operand to be transferred to EU 10122. CLQF clears and "EU 10122's OPB 20322 Full" condition in FU 10120. CLQF indicates, thereby, that there is room in OPB 20322 to receive operands. If FU 10120 is in a "EU 10122's OPB 20322 Full" condition and further operand is required to be transferred to EU 10122, FU 10120 will remain in State M1 until CLQF is asserted.

40 At the beginning of execution of a particular SOP, FU 10120 may transfer two operands to OPB 20322 without "EU 10122's OPB 20322 Full" condition occurring. This is because EU 10122 is idle at the beginning of an SOP execution and generally immediately unloads a first operand from OPB 20322 before a second operand arrives.

45 Control signal Job Processor Operand (JPOP) provided from FU 10120 must be non-asserted for operands to be transferred from MEM 10112 to OPB 20322 through MOD Bus 10144. This is the normal condition of JPOP. If JPOP is asserted, OPB 20322 is loaded with data from JPD Bus 10142. Data is strobed into OPB 20322 from JPD Bus 10142 by control signals M1CPT and JPOP. Operands read from MEM 10112, however, are transferred into OPB 20322 through MOD Bus 10144 when MEM 10112 asserts DAVEB to indicate that valid data from MEM 10112 is available on MOD Bus 10144. DAVEB is also utilized to strobe data on MOD Bus 10144 into OPB 20322. If control signal ZFILL from MEM 10112 is asserted at this point, ZFILL is interpreted during integer operand operations to indicate that those operands are unsigned and

EP 0 067 556 B1

should be left zero filled, rather than sign extended. If data is being provided from JPD Bus 10142 rather than from MEM 10112, that is if JPOP is asserted, bit 11 of current EUDP may be utilized to perform the same function as ZFILL during loading of OPB 20322 from MOD Bus 10144.

5 Loading of OPB 20322 is controlled, in part, by bits 9 and 10 of EUDPs provided from FU 10120 through EUDIS Bus 20206. Bit 9 indicates length of a first operand while bit 10 indicates length of a second operand. Operand length, together with operand type specified in address portion of a EUDP, determines how a particular operand is unloaded from OPB 20322 and transferred into MULT 20314 and EXP 20316.

10 At this point, both COMQ 20342 and OPB 20322 have been loaded with, respectively, EUDPs and operands. It should be noted that operands are generally not transferred into OPB 20322 before a corresponding EUDP is loaded into COMQ 20342. Operands and EUDPs may, however, be simultaneously transferred into EU 10122. If other operands are required for a particular operation, those operands are loaded into OPB 20322 as described above.

15 c.c.c. Storeback (Fig. 262)

Referring to Fig. 262, a diagramic representation of a storeback, or transfer, of results to MEM 10112 from EU 10122 and handshaking performed therein is shown. When a final result of a EU 10122 operation is available, EU 10122 asserts control signal Data Ready (DRDY). FU 10120 thereupon responds with control signal Transfer to JPD Bus 10142 (XJPD), which gates EU 10122's result onto JPD Bus 10142. In normal operation, that is execution of SOPs, FU 10120 causes EU 10122's result to be stored back into a destination in MEM 10112, as selected by a physical descriptor provided from FU 10120. Alternately, a result may be transferred into FU 10120, 32 bits, or one word, at a time.

20 FU 10120 may, as described above and described further below, check EU 10122 test conditions during storeback of results. FU 10120 generates control signal Transfer Complete (XFRC) once the storeback operation is completed. XFRC also indicates to EU 10122 that EU 10122's results and test conditions have been accepted by FU 10120, so that EU 10122 need no longer assert these results and test conditions.

d.d.d. Test Conditions (Fig. 263)

25 Referring to Fig. 263, a diagramic representation of checking of EU10122 test conditions by FU 10120, and handshaking therein, is shown. As previously described, test results indicating certain conditions and operations of EU 10122 are sampled and stored in TSTCOND 20384 and may be examined by FU 10120. When DRDY is asserted by EU 10122, FU 10120 may select, for example, one of 8 EU 10122 conditions to test, as well as transferring results as described above. EU 10122 conditions which may be tested by FU 10120 are listed and described below. Such conditions, as whether a final result is positive, negative, or zero, may be checked in order to facilitate conditional branching of FU 10120 operations as previously described. FU 10120 specifies a condition to be tested through Test Condition Select signals (TEST(24)). FU 10120 asserts control signal EU Test Enable (EUTESTEN) to EU 10122 to gate the selected test condition. That selected test condition then appears as Data Signal Test Condition (TC) from EU 10122 to FU 10120. A TC of logic 1 may, for example, indicate that the selected condition is false while a TC of logic 0 may indicate that the selected condition is true. FU 10120 indicates that FU 10120 has sensed the requested test condition, and that the test condition need no longer be asserted by EU 10122, by asserting control signal XFRC.

35 e.e.e. Exception Checking (Fig. 264)

Referring to Fig. 264, a diagramic representation of exception checking of EU 10122 exceptions by FU 10120, and handshaking therein, is shown. As previously described, any EU 10122 exception conditions may be checked by FU 10120 as FU 10120 is initiating storeback of EU 10122 results. Exception checking may detect, for example, attempted division by zero, floating point exponent underflow or overflow, or a container size fault. An attempted division by zero or floating point underflow or overflow may be checked before storeback, that is without specific request by FU 10120.

40 As previously described, a container size fault is detected by CONSIZE 20352 by comparing length of result with size of destination container in MEM 10112. Container size exception checking occurs during store back of EU 10122 results, that is while FU 10120 is in State SB. Container size is automatically performed by EU 10122 hardware, that is by CONSIZE 20352, only on results of less than 33 bits length. Size checking of larger results, that is larger integers and BCD results, is performed by a microcode routine, using CONSIZE 20352's output, as transfer of such larger results is executed as string transfer. It is unnecessary to perform container size check for either single or double precision floating point results as these data types always occupy either 32 or 64 bits. Destination container size is provided to CONSIZE 20352 through LENGTH Bus 20226.

45 Control signal Length to Memory AON or Random Signals (LMAONRS) is generated by FU 10120 from Type field of the logical descriptor corresponding to a particular EU 10122 result. LMAONRS indicates that the results data type is an unsigned integer. LMAONRS determines the manner in which a required container size of the EU 10122 result is determined. After receiving this information from LMAONRS, EU 10122 determines whether destination container size in MEM 10112 is sufficiently large to contain the EU

EP 0 067 556 B1

10122 result. If that destination container size is not sufficiently large, a container size fault is detected by CONSIZE 20352, or through an EU 10122 microinstruction sequence.

Container size faults, as well as division by zero and exponent underflow and overflow faults, are signaled to FU 10120 when FU 10120 asserts control signal Check Size (CKSIZE). At this time, EU 10122 asserts control signal Exception (EXCPT) if any of the above faults has occurred. If a fault has occurred, an Event request to FU 10120 results. When an Event request is honored by FU 10120, FU 10120 may interrupt EU 10122 and dispatch EU 10122 to a microinstruction routine that transfers those exception conditions onto JPD Bus 10142. If a container size fault has caused that exception condition, EU 10122 may transfer to FU 10120 the required container size through JPD Bus 10142.

f.f.f. Idle Routine

Finally, when a current EU 10122 operation is completed, EU 10122 goes into an Idle loop microinstruction routine. If necessary, FU 10120 may assert control signal Excute Unit Abort (EUABORT) to force EU 10122 into Idle loop microinstruction routine until EU 10122 is required for further operations.

g.g.g. EU 10122 Stack Mechanism (Figs. 265, 266, 267)

As previously described, EU 10122 may perform either of two classes of operations. First, EU 10122 may perform arithmetic operations in execution of SOPs of user's programs. Second, EU 10122 may operate as an arithmetic calculator assisting operation of FU 10120's internal mechanisms and operations, referred to as kernel operations.

In kernel operation, EU 10122 acts as an arithmetic calculator for FU 10120 during address generation, address translation, and other kernel functions. In kernel mode, EU 10122 is executing microinstruction sequences at request of FU 10120 kernel microinstruction sequences, rather than at request of an SOP. In general, these kernel operations are vital to operation of JP 10114. FU 10120 may interrupt EU 10122 operations with regard to SOPs and initiate EU 10122 microinstruction sequences to perform kernel operations.

When interrupted, EU 10122 saves EU 10122's current operating state in a one level deep stack. EU 10122 may then accept an EUDP from that portion of COMQ 20342 utilized to receive and store EUDPs regarding FU 10120's and EU 10122's internal, or kernel, operations. When requesting kernel operations by EU 10122, FU 10120 generally transfers operands to OPB 20322 through JPD Bus 10142, and receives EU 10122 final results through JPD Bus 10142. Operands may also be provided to EU 10122 through MOD Bus 10144. After EU 10122 has completed a requested kernel operation, EU 10122 reloads operating state from its internal stack and continues normal operation from the point normal operation was interrupted.

Should another interrupt from FU 10120 occur while a prior interrupt is being executed, EU 10122 moves current state and data, that is of first interrupt, to MEM 10112. EU 10122 requests FU 10120 store state and date of first interrupt in MEM 10112 by requesting an "EU 10122 Stack Overflow" Event. EU 10122's "normal" state, that is state and data pertaining to the operation EU 10122 is executing at time of occurrence of first interrupt, is stored in an EU 10122 internal stack and remains there. EU 10122 then begins executing second interrupt. When EU 10122 has completed operations for second interrupt, state from first interrupt is reloaded from MEM 10112 by EU 10122 requesting a "EU 10122 Stack Underflow" Event to FU 10120. EU 10122 then completes execution of first interrupt and reloads state and resumes execution of normal operation, that is the operation being executed before the first interrupt.

EU 10122 is therefore capable of handling interrupts from FU 10120 during two circumstances. First interrupt circumstance is comprised of interrupts occurring during normal operation, that is while executing SOPs of user's programs. Second circumstance arises when interrupts occur during kernel operations, that is during execution of microinstruction sequences for handling interrupts. EU 10122 operation will be described next below for each of these circumstances, and in the order referred to.

Referring to Fig. 265, a diagramic representation of EU 10122's stack mechanisms, previously described, is shown. Those portions of EU 10122's stack mechanisms residing within EU 10122 are comprised of EU 10122's Current State Registers (EUCSRs) 26510 and EU 10122's Internal Stack (EUIS) 26512. EUCSR 26510 is comprised of EU 10122's internal registers which contain data and state of current EU 10122 operation. EUCSR 26510 may be comprised, for example, of mCRD 20346, registers of TSTINT 20320, and the previously described registers within MULT 20314 and EXP 20316.

State and data contained in EUCSR 26510 is that of the operation currently being executed by EU 10122. This current state may, for example, be that of a SOP currently being executed by EU 10122, or that of an interrupt, for example a fourth interrupt of a nested sequence of interrupts, requested by FU 10120.

EUIS 26512 is comprised of certain registers of MULTRF 20350 and EXPRF 20380. EUIS 26512 is utilized to store and save current state of an SOP operation currently being executed by EU 10122 and which has been interrupted. State and data of that SOP operation will remain stored in EUIS 26512 regardless of the number of interrupts which may occur on a nested sequence of interrupts requested by FU 10120. State and data of the interrupted SOP operation will be returned from EUIS 26512 to EUCSR 26510 when all interrupts have been completed.

Final portion of EU 10122's stack mechanism is that portion of EU 10122's internal stack (EUES) 26514

EP 0 067 556 B1

residing in MEM 10112. EUES 26514 is comprised of certain MEM 10112 address locations used to store state and data of successive interrupt operations of sequences of nested interrupts. That is, if a sequence of four interrupts is requested by FU 10120, state and data of fourth interrupt will reside in EUCSR 26510 while state and data of first, second, and third interrupts have been transferred, in sequence, into EUES 26514. In this respect, and as previously described operation of EU 10122's stack mechanisms is similar to that of, for example, MIS 10368 and SS 10336 previously described with reference to Fig. 103.

As described above, an interrupt may be requested of EU 10122 by FU 10120 either during EU 10122 normal operation, that is during execution of SOPs by EU 10122, or while EU 10122 is executing a previous interrupt requested by FU 10120. Operation of EU 10122 and FU 10120 upon occurrence of an interrupt during EU 10122 normal operation will be described next below.

Referring to Fig. 266, a diagramic representation of handshaking between EU 10122 and FU 10120 during an interrupt of EU 10122 while EU 10122 is operating in normal mode is shown and should be referred to in conjunction with Fig. 265. For purposes of the following discussions, interrupts of EU 10122 operations by FU 10120 are referred to as nanointerrupts to distinguish from interrupts internal to FU 10120.

FU 10120 interrupts normal operation of EU 10122 by assertion of control signal Nano-Interrupt (NINTP) during State M0 of FU 10120 operation. NINTP may be masked by EU 10122 during certain critical EU 10122 operations, such as arithmetic operations. If NINTP is masked by EU 10122, FU 10120 will remain in State NW until EU 10122 acknowledges the interrupt.

Upon receiving NINTP from FU 10120, EU 10122s transfers state and data of current SOP operation from EUCSR 26510 to EUIS 26512. EU 10122 then asserts control signal Nano-Interrupt Acknowledge (NIACK) to FU 10120 to acknowledge availability of EU 10122 to accept a nanointerrupt. FU 10120 will then enter State M1 and place an EUDP on EUDIS Bus 20206. Loading of COMQ 20342 then proceeds as previously described, with EU 10122 loading nanointerrupt EUDPs into the appropriate registers of COMQ 20342. COMQ 20342 is loaded as previously described and, if JPOP is asserted, data transferred into OPB 20322 from JPD Bus 10142. If JPOP is not asserted, data is taken into OPB 20322 from MOD Bus 10144. EU 10122 then proceeds to execute the required nanointerrupt operation and storing back of results and checking of test conditions proceeds as previously described for EU 10122 normal operation. In general, exception checking is not performed. When EU 10122 has completed execution of the nanointerrupt operation, EU 10122 transfers state and data of the interrupted SOP operation from EUIS 26512 to EUCSR 26510 and resumes execution of that SOP. At this point, EU 10122 asserts control signal Nano-Interrupt Trap Enable (NITE). NITE is received and tested by FU 10120 to indicate end of nanointerrupt processing.

Referring to Fig. 267, a diagramic representation of interfaces between EU 10122, FU 10120, and MEM 10112 during nested, or sequential, EU 10122 interrupts for kernel operations, and handshaking therein, is shown. During the following discussion, it is assumed that EU 10122 is already processing a nanointerrupt for a kernel operation submitted to EU 10122 by FU 10120. FU 10120 may then submit a second, third, or fourth, nanointerrupt to EU 10122 for a further kernel operation. FU 10120 will assert NINTP to request a nanointerrupt of EU 10122. EU 10122's normal mode state and data from a previously executing SOP operation has been stored in, and remains in, EUIS 26512. Current state and data of currently executing nanointerrupt operation in EUCSR 26510 will be transferred to EUES 26514 in MEM 10112 to allow initiation of pending nanointerrupt. EU 10122 will at this time assert NIACK and control signal Execute Unit Event (EXEVT). EXEVT to FU 10120 informs FU 10120 that an EU 10122 Event has occurred, specifically, and in this case, EXEVT requests FU 10120 service of an EU 10122 Stack Overflow. FU 10120 is thereby trapped to an "EU 10122 Stack Overflow" Event Handler microinstruction sequence. This handler transfers current state and data of interrupted nanointerrupt previously executing in EU 10122 into EUES 26514. State and data of interrupted nanointerrupt is transferred to EUES 26514, one 32 bit word at a time. FU 10120 asserts control signals XJPD to gate each of these state and data words onto JPD Bus 10142 and controls transfer of these words into EUES 26514.

Processing of new nanointerrupt proceeds as described above with reference to interrupts occurring during normal operation. If any subsequent nanointerrupts occur, they are handled in the same manner as just described; FU 10120 signals a nanointerrupt to FU 10120, current EU 10122 state and data is saved by FU 10120 in EUES 26514, and new nanointerrupt is processed. After a nested nanointerrupt, that is a nanointerrupt of a sequence of nanointerrupts, has been serviced, EU 10122 asserts control signal EU 10122 Trap (ETRAP) to FU 10120 to request a transfer of a previous nanointerrupt's state and data from EUES 26514 to EUCSR 26510. FU 10120 will retrieve that next previous nanointerrupt state and data from EUES 26514 through MOD Bus 10144 and will transfer that data and state onto JPD Bus 10142. This state and data is returned, one 32 bit word at a time, and is strobed into EU 10122 by JPOP from FU 10120. Processing of that prior nanointerrupt will then resume. The servicing of successively prior nanointerrupts will continue until all previous nanointerrupts have been serviced. Original state and data of EU 10122, that is that of SOP operation which was initially interrupted, is then returned to EUCSR 26510 from EUIS 26512 and execution of that SOP resumed. At this time, EU 10122 asserts NITE to indicate end of EU 10122 kernel operations in regard to nanointerrupts.

Having described structure and operation of EU 10122, FU 10120 and MEM 10112, with respect to servicing of kernel operation nanointerrupts by EU 10122, loading of EU 10122's EUSITT 20344 with microinstruction sequences will be described next below.

h.h.h.h Loading of Execute Unit S-Interpreter Table 20344 (Fig. 268)

Referring to Fig. 268, a diagrammatic representation of interface and handshaking between EU 10122, FU 10120, MEM 10112, and DP 10118 during loading of microinstructions into EUSITT 20344 is shown. As previously described, EUSITT 20344 contains all microinstructions required for control of EU 10122 in
 5 executing kernel nanointerrupt operations and in executing arithmetic operations in response to SOPs of user's programs. EUSITT 20344 may store microinstruction sequences for interpreting arithmetic SOPs of user's programs for, for example, up to 4 different S-Language Dialects. In general, a capacity of storing
 10 microinstruction sequences for arithmetic operations in up to 4 S-Language Dialects is sufficient for most requirements, so that EUSITT 20344 need be loaded with microinstruction sequences only at initialization of CS 10110 operation. Should microinstruction sequences for arithmetic operations of more than 4 S-Language Dialects be required, those microinstruction sequences may be loaded into EUSITT 20344 in the manner as will be described below.

As previously described, a portion of the microinstructions stored in EUSITT 20344 is contained in Read Only Memories and is thus permanently stored in EUSITT 20344. Microinstruction sequences
 15 permanently stored in EUSITT 20344 are, in general, those required for execution of kernel operations. Microinstruction sequences permanently stored in EUSITT 20344 include those used to assist in writing other EU 10122 microinstruction sequences into EUSITT 20344 as required. Certain microinstruction sequences are stored in a Random Access Memory, referred to as the Writeable Control Store (WCS) portion of EUSITT 20344, and include these for interpreting arithmetic operation SOPs of various S-
 20 Language Dialects.

Writing of microinstruction sequences into EU 10122 is initialized by forcing EU 10122 into an Idle state. Initialization of EU 10122 is accomplished by FU 10120 asserting EUABORT or by DP 10118 asserting control signal clear (CLEAR). Either EUABORT or CLEAR will clear a current operation of EU 10122 and force
 25 EU 10122 into Idle state, wherein EU 10122 waits for further EUDPs provided from FU 10120. FU 10120 then dispatches a EUDP initiating loading of EUSITT 20344 to EU 10122 through EUDIS Bus 20206. Load EUSITT 20344 EUDP specifies starting address of a two step microinstruction sequence in the PROM portion of EUSITT 20344. This two step microinstruction sequence first loads zeros into SCAG 25536, which as previously described provides read and write addresses to EUSITT 20344. EUSITT 20344 load
 30 microinstruction sequence then reads a microinstruction from EUSITT 20344 to mCRD 20346. This microinstruction specifies conditions for handshaking operations with FU 10120 so that loading of EUSITT 20344 may begin. At this time, and from this microinstruction word, EU 10122 asserts control signal DRDY to FU 10120 to indicate that EU 10122 is ready to accept EUDPs from FU 10120 for directing loading of EUSITT 20344. This initial microinstruction also generates a write enable control signal for the WCS portion of EUSITT 20344, inhibits loading of mCRD 20346 from EUSITT 20344, and inhibits normal loading
 35 operations of NXTR 25540 and SCAG 25536. This first microinstruction also directs NASS 25526 to accept address inputs from SCAG 25536 and, finally, causes NITE to FU 10120 to be asserted to unmask nanointerrupts from FU 10120.

FU 10120 then generates a read request to MEM 10112, and MEM 10112 transfers a first 32 bit word of a
 40 EU 10122 microinstruction word onto JPD Bus 10142. Each such 32 bit word from MEM 10112 comprises one half of a 64 bit microinstruction word of EU 10122. When FU 10120 receives DRDY from EU 10122, FU 10120 generates control signal Load Writeable Control Store (LDWCS). LDWCS in turn transfers a 32 bit word on JPD Bus 10142 into a first address of the WCS portion of EUSITT 20344. A next 32 bit half word of a EU 10122 microinstruction word is then read from MEM 10112 through JPD Bus 10142 and transferred into the second half of that first address within the WCS portion of EUSITT 20344. The address in SCAG 25536
 45 is then incremented to select a next address within EUSITT 20344 and the process just described repeated automatically, including generation of DRDY and LDWCS, until loading of EUSITT 20344 is completed.

After loading of EUSITT 20344 is completed, the loading process is terminated when FU 10120 asserts NINTP, or DP 10118 asserts Control Signal Load Complete (LOADCR). Either NINTP or LOADCR releases control of operation of NAG 20340 to allow EU 10122 to resume normal operation.

The above descriptions have described structure and operation of EU 10122, including: execution of various arithmetic operations utilizing various operand formats; operation of EU 10122, FU 10120, and MEM 10112 with regard to handshaking; loading of EUDPs and operands; storeback of results; checking of test conditions and exceptions; EU 10122 Stack Mechanisms during normal and kernel operations; and loading of EU 10122 microinstruction sequences into EUSITT 20344. IOS 10116 and DP 10118 will be
 55 described next below, in that order.

D. I/O System 10116 (Figs. 204, 206, 269)

Referring to Fig. 204, a partial block diagram of IOS 10116 is shown. As previously described, IOS 10116
 60 operates as an interface between CS 10110 and the external world, for example, ED 10124. A primary function of IOS 10116 is the transfer of data between CS 10110, that is MEM 10112, and the external world. In addition to performing transfers of data, IOS 10116 controls access between various data sources and sinks of ED 10124 and MEM 10112. As previously described, IOS 10116 directly addresses MEM 10112's physical address space to write data into or read data from MEM 10112. As such, IOS 10116 also performs
 65 address translation, a mapping operation required in transferring data between MEM 10112's physical

address space and address spaces of data sources and sinks in ED 10124.

As shown in Fig. 204, IOS 10116 includes Data Mover (DMOVR) 20410, Input/Output Control Processor (IOCP) 20412, and one or more data channel devices. IOS 10116's data channel devices may include ECLIPSE® Burst Multiplexer Channel (EBMC) 20414, NOVA Data Channel (NDC) 20416, and other data channel devices as required for a particular configuration of a CS 10110 system. IOCP 20412 controls and directs transfer of data between MEM 10112 and ED 10124, and controls and directs mapping of addresses between ED 10124 and MEM 10112's physical address space. IOCP 20412 may be comprised, for example, of a general purpose computer, such as an ECLIPSE® M600 computer available from Data General Corporation of Westboro, Massachusetts.

EBMC 20414 and NDC 20416 comprise data channels through which data is transferred between ED 10124 and IOS 10116. EBMC 20414 and NDC 20416 perform actual transfers of data to and from ED 10124, under control of IOCP 20412, and perform mapping of ED 10124 addresses to MEM 10112 physical addresses, also under control of IOCP 20412. EBMC 20414 and NDC 20416 may respectively be comprised, for example, of an ECLIPSE® Burst Multiplexer Data Channel and a NOVA® Data Channel, also available from Data General Corporation of Westboro, Massachusetts.

DMOVR 20410 comprises IOS 10116's interface to MEM 10112. DMOVR 20410 is the path through which data and addresses are transferred between EBMC 20414 and NDC 20416 and MEM 10112. Additionally, DMOVR 20410 controls access between EBMC 20414, NDC 20416, and other IOS 10116 data channels, and MEM 10112.

ED 10124, as indicated in Fig. 204, may be comprised of one or more data sinks and sources. ED 10124 data sinks and sources may include commercially available disc drive units, line printers, communication lengths, tape units, and other computer systems, including other CS 10110 systems. In general, ED 10124 may include all such data devices as are generally interfaced with a computer system.

a. I/O System 10116 Structure (Fig. 204)

Referring first to the overall structure of IOS 10116, data input/output of ECLIPSE® Burst Multiplexer Channel Adapter and Control Circuitry (BMCAC) 20418 of EBMC 20414 is connected to bi-directional BMC Address and Data (BMCAD) Bus 20420. BMCAD Bus 20420 in turn is connected to data and address inputs and outputs of data sinks and sources of ED 10124.

Similarly, data and address inputs and outputs of NOVA® Data Channel Adapter Control Circuits (NDCAC) 20422 in NDC 20416 is connected to bi-directional NOVA® Data Channel Address and Data (NDCAD) Bus 20424. NDCAD Bus 20424 in turn is connected to address and data inputs and outputs of data sources and sinks of ED 10124. BMCAD Bus 20420 and NDCAD Bus 20424 are paths for transfer of data and addresses between data sinks and sources of ED 10124 and IOS 10116's data channels and may be expanded as required to include other IOS 10116 data channel devices and other data sink and source devices of ED 10124.

Within EBMC 20414, bi-directional data input and output of BMCAC 20418 is connected to bi-directional input and output of BMC Data Buffer (BMCDB) 20426. Data inputs and data outputs of BMCDB 20426 are connected to, respectively, Data Mover Output Data (DMOD) Bus 20428 and Data Mover Input Data (DMID) Bus 20430. Address outputs of BMCAC 20418 are connected to address inputs of Burst Multiplexer Channel Address Translation Map (BMCATM) 20432 and address outputs of BMCATM 20432 are connected onto DMID Bus 20430. A bi-directional control input and output of BMCATM 20432 is connected from bi-directional IO Control Processor Control (IOCP) Bus 20434.

Referring to NDC 20416, as indicated in Fig. 204 data inputs and outputs of NDCAC 20422 are connected, respectively, from DMOD Bus 20428 and to DMID Bus 20430. Address outputs of NDCAC 20422 are connected to address inputs of NOVA® Data Channel Address Translation Map (NDCATM) 20436. Address outputs of NDCATM 20436 are, in turn, connected onto DMID Bus 20430. A bi-directional control input and output of NDCATM 20436 is connected from IOCP Bus 20434.

Referring to IOCP 20412, a bi-directional control input and output of IOCP 20412 is connected from IOCP Bus 20434. Address and data output of IOCP 20412 is connected to NDCAD Bus 20424. An address output of IOCP Address Translation Map (IOCPATM) 20438 within IOCP 20412 is connected onto DMID Bus 20430. Data inputs and outputs of IOCP 20412 are connected, respectively, to DMOD Bus 20428 and DMID Bus 20430. A bi-directional control input and output of IOCP 20412 is connected to a bi-directional control input and output of DMOVR 20410.

Referring finally to DMOVR 20410, DMOVR 20410 includes Input Data Buffer (IDB) 20440, Output Data Buffer (ODB) 20442, and Priority Resolution and Control (PRC) 20444. A data and address input of IDB 20440 is connected from DMID Bus 20430. A data and address output of IDB 20440 is connected to IOM Bus 10130 to MEM 10112. A data output of ODB 20442 is connected from MIO Bus 10129 from MEM 10112, and a data output of ODB 20442 is connected to DMOD Bus 20428. Bi-directional control inputs and outputs of IDB 20440 and ODB 20442 are connected from bi-directional control inputs and outputs of PRC 20444. A bi-directional control input and output of PRC 20444 is connected from a bi-directional control input and output of IOCP 20412 as described above. Another bi-directional control input and output of PRC 20444 is connected to and from IOMC Bus 10131 and thus from a control input and output of MEM 10112. Having described overall structure of IOS 10116, operation of IOS 10116 will be described next below.

b. I/O System 10116 Operation (Fig. 269)

1. Data Channel Devices

Referring first to EBMC 20414, BMCAC 20418 receives data and addresses from ED 10124 through BMCAD Bus 20420. BMCAC 20418 transfers data into BMCDB 20426, where that data is held for subsequent transmission to MEM 10112 through DMOVR 20410, as will be described below. BMCAC 20418 transfers addresses received from ED 10124 to BMCATM 20432. BMCATM 20432 contains address mapping information correlating ED 10124 addresses with MEM 10112 physical addresses. BMCATM 20432 thereby provides MEM 10112 physical addresses corresponding to ED 10124 addresses provided through BMCAC 20418.

When, as will be described further below, EBMC 20414 is granted access to MEM 10112 to write data into MEM 10112, data stored in BMCDB 20426 and corresponding addresses from BMCATM 20432 are transferred onto DMID Bus 20430 to DMOVR 20410. As will be described below, DMOVR 20410 then writes that data into those MEM 10112 physical address locations. When data is to be read from MEM 10112 to ED 10124, data is provided by DMOVR 20410 on DMOD Bus 20428 and is transferred into BMCDB 20426. BMCAC 20418 then reads that data from BMCDB 20426 and transfers that data onto BMCAD Bus 20420 to ED 10124. During transfers of data from MEM 10112 to ED 10124, MEM 10112 does not provide addresses, to be translated into ED 10124 addresses to accompany that data. Instead, those addresses are generated and provided by BMCAC 20418.

NDC 20416 operates in a manner similar to that of EBMC 20414 except that data inputs and outputs of NDCAC 20422 are not buffered through a BMCDB 20426.

As previously described, MEM 10112 has capacity to perform block transfers, that is sequential transfers of four 32 bit words at a time. In general, such transfers are performed through EBMC 20414 and are buffered through BMCDB 20426. That is, BMCDB 20426 allows single 32 bit words to be received from ED 10124 by EBMC 20414 and stored therein until a four word block has been received. That block may then be transferred to MEM 10112. Similarly, a block may be received from MEM 10112, stored in BMCDB 20426, and transferred one word at a time to ED 10124. In contrast, NDC 20416 may generally be utilized for single word transfers.

As indicated in Fig. 204, EBMC 20414, NDC 20416, and each data channel device of IOS 10116 each contain an individual address translation map, for example BMCATM 20432 in EBMC 20414 and NDCATM 20436 in NDC 20416. Address translation maps stored therein are effectively constructed and controlled by IOCP 20412 for each data channel device. IOS 10116 may thereby provide an individual and separate address translation map for each IOS 10116 data channel device. This allows IOS 10116 to insure that no two data channel devices, nor two groups of data sinks and sources in ED 10124, will mutually interfere by writing into and destroying data in a common area of MEM 10112 physical address space. Alternately, IOS 10116 may generate address translation maps for two or more data channel devices wherein those maps share a common, or overlapping, area of MEM 10112's physical address space. This allows data stored in MEM 10112 to be transferred between IOS 10116 data channel devices through MEM 10112, and thus to be transferred between various data sink and source devices of ED 10124. For example, a first ED 10124 data source and a first IOS 10116 data channel may write data to be operated upon into a particular area of MEM 10112 address space. The results of CS 10110 operations upon that data may then be written into a common area shared by that first data device and a second data device and read out of MEM 10112 to a second ED 10124 data sink by that second data channel device. Individual mapping of IOS 10116's data channel devices thereby provides total flexibility in partitioning or sharing of MEM 10112's address space through IOS 10116.

2. I/O Control Processor 20412

As described above, IOCP 20412 is a general purpose computer whose primary function is overall direction and control of data transfer between MEM 10112 and ED 10124. IOCP 20412 controls mapping of addresses between IOS 10116's data channel devices and MEM 10112 address space. In this regard, IOCP 20412 generates address translation maps for IOS 10116's data channel devices, such as EBMC 20414 and NDC 20416. IOCP 20412 loads these address translation maps into and controls, for example, BMCATM 20432 of EBMC 20414 and NDCATM 20436 and NDC 20416 through IOCP Bus 20434. IOCP 20412 also provides certain control functions to DMOVR 20410, as indicated in Fig. 204. In addition to these functions, IOCP 20412 is also provided with data and addressing inputs and outputs. These data addressing inputs and outputs may be utilized, for example, to obtain information utilized by IOCP 20412 in generating and controlling mapping of addresses between IOS 10116's data channel devices and MEM 10112. Also, these data and address inputs and outputs allow IOCP 20412 to operate, in part, as a data channel device. As previously described, IOCP 20412 has data and address inputs and outputs connected from and to DMID Bus 20430 and DMOD Bus 20428. IOCP 20412 thus has access to data being transferred between ED 10124 and MEM 10112, providing IOCP 20412 with direct access to MEM 10112 address space. In addition, IOCP 20412 is provided with control and address outputs to NDCAD Bus 20424, thus allowing IOCP 20412 partial control of certain data source and sink devices in ED 10124.

3. Data Mover 20410 (Fig. 269)

a.a. Input Data Buffer 20440 and Output Data Buffer 20442

As described above, DMOVR 20410 comprises an interface between IOS 10116's data channels and MEM 10112. DMOVR 20410 performs actual transfer of data between IOS 10116's data channel devices and MEM 10112, and controls access between IOS 10116's data channel devices and MEM 10112. IDB 20440 and ODB 20442 are data and address buffers allowing asynchronous transfer of data between IOS 10116 and MEM 10112. That is, ODB 20442 may accept data from MEM 10112 as that data becomes available and then hold that data until an IOS 10116 data channel device, for example EBMC 20414, is ready to accept that data. IDB 20440 accepts data and MEM 10112 physical addresses from IOS 10116's data channel devices. IDB 20440 holds that data and addresses for subsequent transmission to MEM 10112 when MEM 10112 is ready to accept data and addresses. IDB 20440 may, for example, accept a burst, or sequence, of data from EBMC 20414 or single data words from NDC 20416 and subsequently provide that data to MEM 10112 in block, or four word, transfers as previously described. Similarly, ODB 20442 may accept one or more block transfers or data from ODB 20442 and subsequently provide that data to NDC 20416 as single words, or to DMID 20430 as a data burst. In addition, as previously described, a block transfer from MEM 10112 may not appear as four sequential words. In such cases, ODB 20442 accepts the four words of a block transfer as they appear on MIO Bus 10129 and assembles those words into a block comprising four sequential words for subsequent transfer to ED 10124.

Transfer of data through IDB 20440 and ODB 20442 is controlled by PRC 20444, which exchanges control signals with IOCP 20412 and has an interface, previously described, to MEM 10112 through IOMC Bus 10131.

b.b. Priority Resolution and Control 20444 (Fig. 269)

As previously described, PRC 20444 controls access between IOS 10116 data channel devices and MEM 10112. This operation is performed by means of a Ring Grant Access Generator (RGAG) within PRC 20444.

Referring to Fig. 270, a diagramic representation of PRC 20444's RGAG is shown. In general, PRC 20444's RGAG is comprised of a Ring Grant Code Generator (RGCG) 26910 and one or more data channel request comparators. In Fig. 269, PRC 20444's RGAG is shown as including ECLIPSE® Burst Multiplexer Channel Request Comparator (EBMCRC) 26912, NOVA® Data Channel Request Comparator (NDCRC) 26914, Data Channel Device X Request Comparator (DCDXRC) 26916, and Data Channel Device Z Request Comparator (DCDZRC) 26918. PRC 20444's RGAG may include more or fewer request comparators as required by the number of data channel devices within a particular IOS 10116.

As indicated in Fig. 269, Request Grant Code (RGC) outputs of RGCG 26910 are connected in parallel to first inputs of EBMCRC 26912, NDCRC 26914, DCDXRC 26916, and DCDZRC 26918. Second inputs of EBMCRC 26912, NDCRC 26914, DCDXRC 26916, and DCDZRC 26918 are connected from other portions of PRC 20444 and receive indications that, respectively, EBMC 20414, NDC 20416, DCDX, or DCDZ has submitted a request for a read or write access to MEM 10112.

Request Grant Outputs (GRANT) of EBMCRC 26912, NDCRC 26914, DCDXRC 26916, and DCDZRC 26918 are in turn connected to other portions of PRC 20444 circuitry to indicate when read or write access to MEM 10112 has been granted in response to a request by a particular IOS 10116 data channel device. When indication of such a grant is provided to those other portions of PRC 20444, PRC 20444 proceeds to generate appropriate control signals to MEM 10112, through IOMC Bus 10131 as previously described, to IDB 20440 and ODB 20442, and to IOCP 20412. PRC 20444's control signals initiate that read or write request to that IOS 10116 data channel device. Grant outputs of EBMCRC 26912, NDCRC 26914, DCDXRC 26916, and DCDZRC 26918 are also provided as inputs to RGCG 26910 to indicate, as described further below, when a particular IOS 10116 has requested and been granted access to MEM 10112.

As indicated in Fig. 269, a diagramic figure above RGCG 26910, RGCG generates a repeated sequence of unique RGCs. Herein indicated as numeric digits 0 to 15. Each RGC identifies, or defines, a particular time slot during which a IOS 10116 data channel device may be granted access to MEM 10112. Certain RGCs are, effectively, assigned to particular IOS 10116 data channel devices. Each such data channel device may request access to MEM 10112 during its assigned RGC identified access slots. For example, EBMC 20414 is shown as being allowed access to MEM 10112 during those access slots identified by RGCs 0, 2, 4, 6, 8, 10, 12, and 14. NDC 20416 is indicated as being allowed access to MEM 10112 during RGC slots 3, 7, 11, and 15. DCDX is allowed access during slots 1 and 9, and DCDZ is allowed access during RGC slots 5 and 13.

As described above, RGCG generates RGCs 0 to 15 in a repetitive sequence. During occurrence of a particular RGC, each request comparator of PRC 20444's RGAG examines that RGC to determine whether its associated data channel device is allowed access during that RGC slot, and whether that associated data channel device has requested access to MEM 10112. If that associated data channel device is allowed access during that RGC slot, and has requested access, that data channel device is granted access as indicated by that request comparator's GRANT output. The request comparators GRANT output is also provided as an input to RGCG 26910 to indicate to RGCG 26910 that access has been granted during that RGC slot.

If a particular data channel device has not claimed and has not been granted access to MEM 10112

during that RGC slot, RGCG 26910 will go directly to next RGC slot. In next RGC slot, PRC 20444's RGAG again determines whether the particular data channel device allowed access during that slot has submitted a request, and will grant access if such a request has been made. If not, RGCG 26910 will again proceed directly to next RGC slot, and so on. In this manner, PRC 20444's RGAG insures that each data channel device of IOS 10116 is allowed access to MEM 10112 without undue delay. In addition, PRC 20444's RGAG prevents a single, or more than one, data channel device from monopolizing access to MEM 10112. As described above, each data channel device is allowed access to MEM 10112 at least once during a particular sequence of RGCs. At the same time, by not pausing within a particular RGC in which no request for access to MEM 10112 has occurred, PRC 20444's RGAG effectively automatically skips over those data channel devices which have not requested access to MEM 10112. PRC 20444's RGAG thereby effectively provides, within a given time interval, more frequent access to those data channel devices which are most busy. In addition, the RGCs assigned to particular IOS 10116 data channel devices may be reassigned as required to adapt a particular CS 10110 to the data input and output requirements of a particular CS 10110 configuration. That is, if EBMC 20414 is shown to require less access to MEM 10112 than NDC 20416, certain RGCs may be reassigned from EBMC 20414 to NDC 20416. Access to MEM 10112 by IOS 10116's data channel devices may thereby be optimized as required.

Having described structure and operation of IOS 10116, structure and operation of DP 10118 will be described next below.

E. Diagnostic Processor 10118 (Figs. 101, 205)

Referring to Fig. 101, as previously described, DP 10118 is interconnected with IOS 10116, MEM 10112, FU 10120, and EU 10122 through DP Bus 10138. DP 10118 is also interconnected, through DPIO Bus 10136, with the external world and in particular with DU 10134. In addition to performing diagnostic and fault monitoring and correction operations, DP 10118 operates, in part, to provide control and display functions allowing an operator to interface with CS 10110. DU 10134 may be comprised, for example, of a CRT and keyboard unit, or a teletype, and provides operators of CS 10110 with all control and display functions which are conventionally provided by a hard console, that is a console containing switches and lights. For example, DU 10134, through DP 10118, allows an operator to exercise control of CS 10110 for such purposes as system initialization and startup, execution of diagnostic processes, fault monitoring and identification, and control of execution of programs. As will be described further below, these functions are accomplished through DP 10118's interfaces with IOS 10116, MEM 10112, FU 10120, and EU 10122.

DP 10118 is a general purpose computer system, for example a NOVA® 4 computer of Data General Corporation of Westboro, Massachusetts. Interface of DP 10118 and DU 10134, and mutual operation of DP 10118 and DU 10134, will be readily apparent to one of ordinary skill in the art. DP 10118's interface and operation, with IOS 10116, MEM 10112, FU 10120, and EU 10122 will be described further next below.

DP 10118, operating as a general purpose computer programmed specifically to perform the functions described above, has, as will be described below, read and write access to registers of IOS 10116, MEM 10112, FU 10120 and EU 10122 through DP Bus 10138. DP 10118 may read data directly from and write data directly into those registers. As will be described below, these registers are data and instruction registers and are integral parts of CS 10110's circuitry during normal operation of CS 10110. Access to these registers thereby allows DP 10118 to directly control or effect operation of CS 10110. In addition, and as also will be described below, DP 10118 provides, in general, all clock signals to all portions of CS 10110 circuitry and may control operation of that circuitry through control of these clock signals.

For purposes of DP 10118 functions, CS 10110 may be regarded as subdivided into groups of functionally related elements, for example DESP 20210 in FU 10120. DP 10118 obtains access to the registers of these groups, and control of clocks therein, through scan chain circuits, as will be described next below. In general, DP 10118 is provided with one or more scan chain circuits for each major functional sub-element of CS 10110.

Referring to Fig. 205, a diagrammatic representation of DP 10118 and a typical DP 10118 scan chain is shown. As indicated therein, DP 10118 includes a general purpose Central Processor Unit, or computer, (DPCPU) 27010. A first interface of DPCPU 27010 is with DU 10134 through DPIO Bus 10136. DPCPU 27010 and DU 10134 exchange data and control signals through DPIO Bus 10136 in the manner to direct operations of DPCPU 27010, and to display the results of those operations through DU 10134.

Associated with DPCPU 27010 is Clock Generator (CLKG) 27012. CLKG 27012 generates, in general, all clock signals used within CS 10110.

DPCPU 27010 and CLKG 27012 are interfaced with the various scan chain circuits of CS 10110 through DP Bus 10138. As described above, CS 10110 may include one or more scan chains for each major sub-element of CS 10110. One such scan chain, for example DESP 20210 Scan Chain (DESPSC) 27014 is illustrated in Fig. 205.

Interface between DPCPU 27010 and CLKG 27012 and, for example, DESPSC 27014 is provided through DP Bus 10138. As indicated in Fig. 205, DESPSC 27014 includes Scan Chain Clock Gates (SCCG) 27016 and one or more Scan Chain Registers (SCRs) 27024.

SCCG 27016 receives clock signals from CLKG 27012 and control signals from DPCPU 27010 through DP Bus 10138. SCCG 27016 in turn provides appropriate clock signals to the various registers and circuits

of, for example, DESP 20210. Clock control signals provided by DPCPU 27010 to SCCG 27016 control, or gate, the various clock signals to these registers and circuits of DESP 20210, thereby effectively allowing DPCPU 27010 to control of DESP 20210.

5 SCRs 27018 to 27024 are comprised of various registers within DESP 20210. For example, SCRs 27018 to 27024 may include the output buffer registers of AONGRF 20232, OFFGRF 20234, LENGRF 20236, output registers of OFFALU 20242 and LENALU 20252, and registers within OFFMUX 20240 and BIAS 20246. Such registers are indicated in the present description, as previously described, by arrows appended to ends of those registers, with a first arrow indicating an input and a second an output. In normal CS 10110 operations, as previously described, SCRs 27018 to 27024 operate as parallel in, parallel out buffer registers through which data and instructions are transferred. SCRs 27018 to 27024 are also capable of operating as shift registers and, as indicated in Fig. 205, are connected together to comprise a single shift register circuit having an input from DPCPU 27010 and an output to DPCPU 27010. Control inputs to SCRs 27018 to 27024 from DPCPU 27010 control operation of SCRs 27018 to 27024, that is whether these registers shall operate as parallel in, parallel out registers, or as shift registers of DESPSC 27014's scan chain. The shift register scan chain comprising SCRs 27018 to 27024 allows DPCPU 27010 to read the contents of SCRs 27018 to 27024 by shifting the content of these registers into DPCPU 27010. Conversely, DPCPU 27010 may write into SCRs 27018 to 27024 by shifting information generated by DPCPU 27010 from DPCPU 27010 and through the shift register scan chain to selected locations within SCRs 27018 to 27024.

20 Scan chain clock generator circuits and scan chain registers of each scan chain circuit within CS 10110 thereby allow DP 10118 to control operation of each major sub-element of CS 10110. For example, to read information from the scan chain registers therein, and to write information into those scan chain registers as required for diagnostic, monitoring, and control functions.

25 Having described structure and operation of each major element of CS 10110, including MEM 10112, FU 10120, EU 10122, IOS 10116, and DP 10118, certain operations of, in particular, FU 10120 will be described further next below. The following descriptions will further disclose operational features of JP 10114, and in particular FU 10120, by describing in greater detail certain operations therein by further describing microcode control of JP 10114.

30 F. CS 10110 Micromachine Structure and Operation (Figs. 270—274)

a. Introduction

35 The preceding descriptions have presented the hardware structures and operation of FU 10120 and EU 10122. The following description will describe how devices in FU 10120, and certain EU 10122 devices, function together as a microprogrammable computer, henceforth termed the FU micromachine. The FU micromachine performs two tasks: it interprets SInS, and it responds to certain signals generated by devices in FU 10120, EU 10122, MEM 10112, and IOS 10116. The signals to which the FU micromachine responds are termed Event signals. In terms of structure and operation, the FU micromachine is characterized by the following:

- 40 — Registers and ALUs specialized for the handling of logical descriptors.
- Registers organized as stacks for invocations of microroutines (microinstruction sequences).
- Mechanisms allowing microroutine invocations by means of event signals from hardware.
- Mechanisms which allow an invoked microroutine to return either to the microinstruction following the one which resulted in the invocation or to the microinstruction which resulted in the invocation.
- 45 — Mechanisms which allow the contents of stack registers to be transferred to MEM 10112, thereby creating a virtual microstack of limitless size.
- Mechanisms which guarantee response to an event signal within a predictable length of time.
- The division of the devices comprising the micromachine into two groups: those devices which may be used by all microcode and those which may be used only by KOS (Kernel Operating System, previously described) microcode.

50 These devices and mechanisms allow the FU micromachine to be used in two ways: as a virtual micromachine and as a monitor micromachine. Both kinds of micromachine use the same devices in FU 10120, but perform different functions and have different logical properties. In the following discussion, when the FU micromachine is being used as a virtual micromachine, it is said to be in virtual mode, and when it is being used as a monitor micromachine, it is said to be in monitor mode. Both modes are introduced here and explained in detail later.

55 When the FU micromachine is being used in virtual mode, it has the following properties:

- It runs on an essentially infinite micromachine stack belonging to a Process 610.
- It can respond to any number of event signals in the M0 cycle (state) of a single microinstruction.
- A page fault may occur on the invocation of any microroutine or on return from any microroutine.
- 60 — When the FU micromachine is in virtual mode, any microroutine may not run to completion, i.e., complete its execution in a predictable length of time, or complete it at all.
- It is executing a Process 610.

65 The last four properties are consequences of the first: Event signals result in invocations, and since the micromachine stack is infinite, there is no limit to the number of invocations. The infinite micromachine stack is realized by placing micromachine stack frames on Secure Stack 10336 belonging to a Process 610,

and the virtual micromachine therefore always runs on a micromachine stack belonging to some Process 610. Furthermore, if the invocation of a microroutine or a return from a microroutine requires micromachine frames to be transferred from Secure Stack 10336 to the FU micromachine, a page fault may result, and Process 610 which is executing the microroutine may be removed from JP 10114, thereby making the time required to execute the microroutine unpredictable. Indeed, if process 610 is stopped or killed, the execution of the microroutine may never finish. As will be seen in descriptions below, the Virtual Processor 612 is the means by which the virtual micromachine gains access to a Process 610's micromachine stack.

When in monitor mode, the FU micromachine has the following properties:

- It has a micromachine stack of fixed size, the stack is always available to the FU micromachine, and it is not associated with a Process 610.
- It can respond to only a fixed number of events during the M0 cycle of a single microinstruction.
- In monitor mode, invocation of a microroutine or return from a microroutine will not cause a page fault.
- Microroutines executing on the FU micromachine when the micromachine is in monitor mode are guaranteed to run to completion unless they themselves perform an action which causes them to give up JP 10114.
- Microroutines executing in monitor mode need not be performing functions for a Process 610.

Again, the remaining properties are consequences of the first: because the monitor micromachine's stack is of fixed size, the number of events to which the monitor micromachine can respond is limited; furthermore, since the stack is always directly accessible to the micromachine, microroutine invocations and returns will not cause page faults, and microroutines running in monitor mode will run to completion unless they themselves perform an action which causes them to give up JP 10114. Finally, the monitor micromachine's stack is not associated with a Process 610's Secure Stack 10336, and therefore, the monitor micromachine can both execute functions for Processes 610 and execute functions (which are related to no Process 610, for example,) the binding and removal of Virtual Processors 612 from JP 10114.

The description which follows first gives an overview of the devices which make up the micromachine, continues with descriptions of invocations on the micromachine and micromachine programming, and concludes with detailed discussions of the virtual and monitor modes and an overview of the relationship between the micromachine and CS 10110 subsystems. The manner in which the micromachine performs specific operations such as SIN parsing, Name resolution, or address translation may be found in previous descriptions of CS 10110 components which the micromachine uses to perform the operations.

b. Overview of Device Comprising FU Micromachine (Fig. 270)

Fig. 270 presents an overview of the devices comprising the micromachine. Fig. 270 is based on Fig. 201, but has been simplified to improve the clarity of the discussion. Devices and subdivisions of the micromachine which appear in Fig. 201 have the numbers given them in that figure. When a device in Fig. 270 appears in two subdivisions, it is shared by those subdivisions.

Fig. 270 has four main subdivisions. Three of them are from Fig. 201: FUCTL 20214, which contains the devices used to select the next microinstruction to be executed by the micromachine, DESP 20210, which contains stack and global registers and ALUs for descriptor processing; and MEMINT 20212, which contains the devices which translate Names into logical descriptors and logical descriptors into physical descriptors. The fourth subdivision, EU Interface 27007, represents those portions of EU 10122 which may be manipulated by FU 10120 microcode.

Fig. 270 further subdivides FUCTL 20214 and MEMINT 20212. FUCTL 20214 has four subdivisions:

- I-Stream Reader 27001, which contains the devices used to obtain SINS and parse them into SOPs and Names.
- SOP Decoder 27003, which translates SOPs into locations in FU microcode (FUSITT 11012), and in some cases EU microcode (EUSITT 20344), which contain the microcode that performs the corresponding SINS.
- Microcode Addressing 27013, which determines the location of the next microinstruction to be executed in FUSITT 11012.
- Register Addressing 27011, which contains devices which generate addresses for GRF 10354 registers.

MEMINT 20212 also has three subdivisions:

- Name Translation Unit 27015, which contains devices which accelerate the translation of Names into logical descriptors.
- Memory Reference Unit 27017, which contains devices which accelerate the translation of logical descriptors into physical descriptors.
- Protection Unit 27019, which contains devices which accelerate primitive access checks on memory references made with logical descriptors.

Fig. 270 also simplifies the bus structure of Fig. 202 by combining LENGTH Bus 20226, OFFSET Bus 20228, and AONR Bus 20230 into a single structure, Descriptor Bus (DB) 27021. In addition, internal bus connections have been reduced to those necessary for explaining the logical operation of the

EP 0 067 556 B1

micromachine. The following discussion first describes those devices used by most microcode executing on FU 10120, and then describes devices used to perform special functions, such as Name translation or protection checking.

5

1. Devices used by Most Microcode

The subdivisions of the micromachine which contain devices used by most microcode are Microcode Addressing 27013, Register Addressing 27011, DESP 20210, and EU Interface 27007. In addition, most microcode uses MOD Bus 10144, JPD Bus 10142, and DB Bus 27021. The discussion begins with the buses
10 and then describes the other devices in the above order.

a.a. MOD Bus 10144, JPD Bus 10142, and DB Bus 27021

MOD Bus 10144 is the only path by which data may be obtained from MEM 10112. Data on MOD Bus
15 10144 may have as its destination Instruction Stream Reader 27001, DESP 20210, or EU Interface 27007. In the first case, the data on MOD Bus 10144 consists of SInS; in the second, it is data to be processed by FU 10120, and in the third, it is data to be processed by EU 10122. In the present embodiment, data to be processed by FU 10120 is generally data which is destined for internal use in FU 10120, for example in Name Cache 10226. Data to be processed by EU 10122 is generally operands represented by Names in
20 SInS.

JPD Bus 10142 has two uses: it is the path by which data returns to MEM 10112 after it has been processed by JP 10114, and it is the path by which data other than logical descriptors moves between the subdivisions of the micromachine. For example, when CS 10110 is initialized, the microinstructions which are loaded into FUSITT 11012 are transferred from MEM 10112 to DESP 20210 via MOD Bus 10144, and
25 from DESP 20210 to FUSITT 11012 via JPD Bus 10142.

DB 27021 is the path by which logical descriptors are transferred in the micromachine. DB 27021 connects Name Translation Unit 27015, DESP 20210, Protection Unit 27019, and Memory Reference Unit 27017. Typically, a logical descriptor is obtained from Name Translation Unit 27015, placed in a register in
30 DESP 20210, and then presented to Protection Unit 27019 and Memory Reference Unit 27017 whenever a reference is made using a logical descriptor. However, DB 27021 is also used to transmit cache entries fabricated in DESP 20210 to ATU 10228, Name Cache 10226 and Protection Cache 10234.

b.b. Microcode Addressing

As discussed here, microcode addressing is comprised of the following devices: Timers 20296, Event
35 Logic 20284, RCWS 10358, BRCASE 20278, mPC 20276, MCW0 20292, MCW1 20290, SITTNAS 20286, and FUSITT 11012. All of these devices have already been described in detail, and they are discussed here only as they affect microcode addressing. Other devices contained in Fig. 202, State Registers 20294, Repeat Counter 20280, and PNREG 20282 are not directly relevant to microcode addressing, and are not discussed
40 here.

As has already been described in detail, devices in Microcode Addressing 27013 are loaded from JPD Bus 10142. The microcode addresses provided by these devices and by FUSDT 11010 are transmitted among the devices and to FUSITT 11012 by CSADR Bus 20204. There are six ways in which the next microcode address may be obtained:

- 45 — Most commonly, the value in mPC 20276 is incremented, by 1 by a special ALU in mPC 20276, thus yielding the address of the microinstruction following the current microinstruction.
- If a microinstruction specifies a call to a microroutine or a branch, the microinstruction contains a literal which an ALU in BRCASE 20278 adds to the value in mPC 20276 to obtain the location of the next microinstruction.
- 50 — If a microinstruction specifies the use of a case value to calculate the location of the next microinstruction, BRCASE 20278 adds a value calculated by DESP 20210 to the value in mPC 20276. The value calculated by DESP 20210 may be obtained from a field of a logical descriptor, thus allowing the micromachine to branch to different locations in microcode on the basis of type information contained in the logical descriptor. On return from an invocation of a microroutine, the location at
55 which execution of the microroutine in which the invocation occurred is to continue is obtained from RCWS 10358.
- At the beginning of the execution of an SInS, the location at which the microcode for the SInS begins is obtained from the SInS's SOP by means of FUSDT 11010.
- 60 — Certain hardware signals cause invocations of microroutines. There are two classes of such signals: Event signals, which Event Logic 20284 transforms into invocations of certain microroutines, and JAM signals, which are translated directly into locations in microcode.

The addresses obtained as described above are transmitted to SITTNAS 20286, which selects one of the addresses as the location of the next microinstruction to be executed and transmits the location to FUSITT 11012. As the location is transmitted to FUSITT 11012, it is also stored in mPC 20276. All addresses
65 except those for Jams are transferred to SITTNAS 20286 via CSADR Bus 20204. Addresses obtained from

JAM signals are transferred by separate lines to SITTNAS 20286.

As will be explained in detail below, microroutine calls and returns also involve pushing and popping micromachine stack frames and saving state contained in MCW1 20290.

Register Addressing 27011 controls access to micromachine registers contained in GRF 10354. As explained in detail below, GRF 10354 contains both registers used for the micromachine stack and global registers, that is, registers that are always accessible to all microroutines. The registers are grouped in frames, and individual registers are addressed by frame number and register number. Register Addressing 27011 allows addressing of any frame and register in the GRs 10360 of GRF 10354, but allows addressing of registers in only three frames of the SR's 10362: the current (top) frame, the previous frame (i.e., the frame preceding the top frame), and the bottom frame, that is, the lowest frame in a virtual micromachine stack which is still contained in GRF 10354. The values provided by Register Addressing 27011 are stored in MCW0 20292. As will be explained in the discussion of microroutine invocations which follows, current and previous are incremented on each invocation and decremented on each return.

c.c. Description Processor 20210 (Fig. 271)

DESP 20210 is a set of devices for storing and processing logical descriptors. The internal structure of DESP 20210's processing devices has already been explained in detail; here, the discussion deals primarily with the structure and contents of GRF 10354. In a present embodiment of CS 10110, GRF 10354 contains 256 registers. Each register may contain a single logical descriptor. Fig. 271 illustrates a Logical Descriptor 27116 in detail. In a present embodiment of CS 10110, a Logical Descriptor 27116 has four main fields:

- RS Field 27101, which contains various flags which are explained in detail below.
- AON Field 27111, which contains the AON portion of the address of the data item represented by the Logical Descriptor 27116.
- OFF Field 27113, which contains the offset portion of the address of the data item represented by Logical Descriptor 27116.
- LEN Field 27115, which contains the length of the data item represented by the Logical Descriptor 27116.

RS Field 27101 has subfields as follows:

- RTD Field 27103 and WTD Field 27105 may be set by microcode to disable certain Event signals provided for debuggers by CS 10110. For details, see a following description of debugging aids in CS 10110.
- FIU Field 27107 contains two bits. The fields are set from information in the Name Table Entry used to construct the Logical Descriptor 27116. The bits determine how the data specified by the Logical Descriptor 27116 is to be justified and filled when it is fetched from MEM 10112.
- TYPE Field 27109's four bits are also obtained from the Name Table Entry used to construct the Logical Descriptor 27116. The field's settings vary from S-Language to S-Language, and are used to communicate S-Language-specific type information to the S-Language's S-Interpreter microcode.

The four fields of a Logical Descriptor 27116 are contained in three separately-accessible fields in a GRF 10354 register: one containing RS Field 27101 and AON Field 27111, one containing OFF Field 27113, and one containing LEN Field 27115. In addition, each GRF 10354 register may be accessed as a whole. GRF 10354 is further subdivided into 32 frames of eight registers each. An individual GRF 10354 register is addressed by means of its frame number and its register number within the frame. In a present embodiment of CS 10110, half of the frames in GRF 10354 belong to SR's 10362 and are used for micromachine stacks, and half belong to GRs 10360 for storing "global information". In SR's 10362, each GRF 10354 frame contains information belonging to a single invocation of a microroutine. As previously explained, Register Addressing 27011 allows addressing of only three GRF 10354 frames in SR's tack 10362, the current top frame in the stack, the previous frame, and the bottom frame. Registers are accessed by specifying one of these three frames and a register number.

The global information contained in GRs 10360, is information which is not connected with a single invocation. There are three broad categories of global information:

- Information belonging to Process 610 whose Virtual Processor 612 is currently bound to JP 10114. Included in this information are the current values of Process 610's ABPs and the pointers which KOS uses to manage Process 610's stacks.
- Information required for the operation of KOS. Included in this information are such items as pointers to KOS data bases which occupy fixed locations in MEM 10112.
- Constants, that is, fixed values required for certain frequently performed operations in FU 10120.

Remaining registers are available to microprogrammers as temporary storage areas for data which cannot be stored in a microroutine's stack frame. For example, data which is shared by several microroutines may best be placed in a GR 10360. Addressing of registers in the GRs 10360 of GRF 10354 requires two values: a value of 0 through 15 to specify the frame and a value of 0 through 7 to specify the register in the frame.

As previously discussed in detail, each of the three components AONP 20216, OFFP 20218, and LENP 20220 of DESP 20210 also contains ALUs, registers, and logic which allows operations to be performed on individual fields of GRF 10354 registers. In particular, OFFP 20218 contains OFFALU 20242, which may be

EP 0 067 556 B1

used as a general purpose 32 bit arithmetic and logical unit. OFFALU 20242 may further serve as a source and destination for JPD Bus 10142, the offset portion of DB 27021, and NAME Bus 20224, and as a destination for MOD Bus 10144. Consequently, OFFALU 20242 may be used to perform operations on data on these buses and to transfer data from one bus to another. For example, when an SIN contains a literal value used in address calculation, the literal value is transferred via NAME Bus 20224 to OFFALU 20242, operated on, and output via the offset portion of DB 27021.

d.d. EU 10122 Interface

FU 10120 specifies what operation EU 10122 is to perform, what operands it is to perform it on, and when it is finished, what is to be done with the operands. FU 10120 can use two devices in EU 10122 as destinations for data, and one device as a source for data. The destinations are COMQ 20342 and OPB 20322. COMQ 20342 receives the location in EUSITT 20344 of the microcode which is to perform the operation desired by the FU 10120. COMQ 20342 may receive the location in microcode either from an FU 10120 microroutine or from an SIN's SOP. In the first case, the location is transferred via JPD Bus 10142, and in the second, it is obtained from EUSDT 20266 and transferred via EUDIS Bus 20206. OPB 20322 receives the operands upon which the operation is to be performed. If the operands come directly from MEM 10112, they are transferred to OPB 20322 via MOD Bus 10144; if they come from registers or devices in FU 10120, they are transferred via JPD Bus 10142.

Result Register 27013 is a source for data. After EU 10122 has completed an operation, FU 10120 obtains the result from Result Register 27013. FU 10120 may then place the result in MEM 10112 or in any device accessible from JPD Bus 10142.

2. Specialized Micromachine Devices

Each of the groups of specialized devices serves one of CS 10110's subsystems. I-Stream Reader 27001 is part of the S-Interpreter subsystem, Name Translation Unit 27015 is part of the Name Interpreter subsystem, Memory Reference Unit 27017 is part of the Virtual Memory Management System, and Protection Unit 27019 is part of the Access Control System. Here, these devices are explained only in the context of the micromachine; for a complete understanding of their functions within the subsystems to which they belong, see previous descriptions of the subsystems.

a.a. I-Stream Reader 27001

I-Stream Reader 27001 reads and parses a stream of SINs (termed the I-Stream) from a Procedure Object 604, 606, 608. The I-Stream consists of SOPs (operation codes), Names, and literals. As previously mentioned, in a present embodiment of CS 10110, the I-Stream read from a given Procedure 602 has a fixed format: the SOPs are 8 bits long and the Names and literals all have a single length. Depending on the procedure, the length may be 8, 12, or 16 bits. I-Stream Reader 27001 parses the I-Stream by breaking it up into its constituent SOPs and Names and passing the SOPs and Names to appropriate parts of the micromachine. I-Stream Reader 27001 contains two groups of devices:

- PC Values 27006, which is made up of three registers which contain locations in the I-Stream. When added to ABP PBP, the values contained in these registers specify locations in Procedure Object 901 containing the Procedure 602 being executed. CPC 20270 contains the location of the SOP or Name currently being interpreted; IPC 20272 contains the location of the beginning of the SIN currently being executed; EPC 20274, finally, is of interest only at the beginning of the execution of an SIN; at that time, it contains the location of the last SIN to be executed.
- Parsing Unit 27005, which is made up of INSTB 20262, PARSER 20264, and PREF 20260. The micromachine uses PREF 20260 to create Logical Descriptors 27116 for the I-Stream, which are then placed on DB Bus 27021 and used in logical memory references. The data returned from these references is placed in INSTB 20262, and parsed by PARSER 20264.

SOPs, Names, and literals obtained by PARSER 20264 are placed on NAME Bus 20224, which connects PARSER 20264, SOP Decoder 27003, Name Translation Unit 27015, and OFFALU 20242.

b.b. SOP Decoder 27003

SOP Decoder 27003 decodes SOPs into locations in FU 10120 and EU 10122 microcode. SOP Decoder 27003 comprises FUSDT 11010, EUSDT 20266, Dialect Register (RDIAL) 24212, and LOPDCODE 24210. FUSDT 11010 are further comprised of FUDISP 24218 and FALG 24220. The manner in which these devices translate SOPs contained in SINs into locations in FUSITT 11012 and EUSITT 20344 has been previously described.

c.c. Name Translation Unit 27015

Name Translation Unit 27015 accelerates the translation of Names into Logical Descriptors 27116. This operation is termed name resolution. It is comprised of two components: NC 10226 and Name Trap 20254. NC 10226 contains copies of information from a Procedure Object 604's Name Table 10350, and thereby makes it possible to translate Names into Logical Descriptors 27116 without referring to Name Table 10350. When a Name is presented to Name Translation Unit 27015, it is latched into Name Trap 20254 for later use by Name Translation Unit 27015 if required. As will be explained in detail later, in the present embodiment,

EP 0 067 556 B1

Name translation always begins with the presentation of a Name to NC 10226. If the Name has already been translated, the information required to construct its Logical Descriptor 27116 may be contained in NC 10226. If there is no information for the Name in NC 10226, Name Resolution Microcode obtains the Name from Name Trap 20254, uses information from Name Table 10350 for the procedure being executed to translate the Name, places the required information in NC 10226, and attempts the translation again. When the translation succeeds, a Logical Descriptor 27116 corresponding to the Name is produced from the information in Name Cache 10115, placed on DB Bus 27021, and loaded into a GRF 10354 register.

d.d. Memory Reference Unit 27017

Memory Reference Unit 27017 performs memory references using Logical Descriptors 27116. Memory Reference Unit 27017 receives a command for MEM 10112 and a Logical Descriptor 27116 describing the data upon which the command is to be performed. In the case of a write operation, Memory Reference Unit 27017 also receives the data being written via JPD Bus 10142. Memory Reference Unit 27017 translates Logical Descriptor 27116 to a physical descriptor and transfers the physical descriptor and the command to MEM 10112 via PD Bus 10146. A Memory Reference Unit 27017 has four components: ATU 10228, which contains copies of information from KOS virtual memory management system tables, and thereby accelerates logical-to-physical descriptor translation; Descriptor Trap 20256, which traps Logical Descriptors 27116, Command Trap 27018, which traps memory commands; and Data Trap 20258, which traps data on write operations. When a logical memory reference is made, a Logical Descriptor 27116 is presented via DB Bus 27021 to ATU 10228, and at the same time, Logical Descriptor 27116 and the memory command are trapped in Descriptor Trap 20256 and Command Trap 27018. On write operations, the data to be written is trapped in Data Trap 20258. If the information needed to form the physical descriptor is present in ATU 10228, the physical descriptor is transferred to MEM 10112 via PD Bus 10146. If the information needed to form the physical descriptor is not present in ATU 10228, an Event Signal from ATU 10228 invokes a microroutine which retrieves Logical Descriptor 27116 from Descriptor Trap 20256 and uses information contained in KOS virtual memory management system tables to make an entry in ATU 10228 for Logical Descriptor 27116. When the microroutine returns, the logical memory reference is repeated using Logical Descriptor 27116 from Descriptor Trap 20256, the memory command from Command Trap 27018, and on write operations, the data in Data Trap 20258. As will be described in detail in the discussion of virtual memory management, if the data referenced by a logical memory reference is not present in MEM 10112, the logical memory reference causes a page fault.

e.e. The Protection Unit 27019

On each logical memory reference, Protection Unit 27019 checks whether the subject making the reference has access rights which allow it to perform the action specified by the memory command on the object being referenced. If the subject does not have the required access rights, a signal from Protection Unit 27019 causes MEM 10112 to abort the logical memory reference. Protection Unit 27019 consists of Protection Cache 10234, which contains copies of information from KOS Access Control System tables, and thereby speeds up protection checking, and shares Descriptor Trap 20256, Command Trap 27018, and Data Trap 20258 with Memory Reference Unit 27017. When a logical memory reference is made, the AON and offset portions of the logical descriptor are presented to Protection Cache 10234. If Protection Cache 10234 contains protection information for the object specified by the AON and offset and the subject performing the memory reference has the required access, the memory reference may continue; if Protection Cache 10234 contains protection information and the subject does not have the required access, a signal from Protection Cache 10234 aborts the memory reference. If Protection Cache 10234 does not contain the required access information, a signal from Protection Cache 10234 aborts the memory reference and invokes a microroutine which obtains the access information from KOS Access Control System tables and places it in Protection Cache 10234. When Protection Cache 10234 is ready, the memory access is repeated, using the logical descriptor from Descriptor Trap 20256, the memory command from Command Trap 27018, and in the case of write operations, the data in Data Trap 20258.

f.f. KOS Micromachine Devices

As mentioned in the above introduction to the micromachine, the devices making up the micromachine may be divided into two classes: those which any microcode written for the micromachine may manipulate, and those which may be manipulated exclusively by KOS microcode. The latter class consists of certain registers in GRs 10360 of GRF 10354, the bottom frame of the portion of the virtual micromachine stack in the stack portion (Stack Registers 10362) of GRF 10354, and the devices contained in Protection Unit 27019 and Memory Reference Unit 27017. Because Protection Unit 27019 and Memory Reference Unit 27017 may be manipulated only by KOS microcode, non-KOS microcode may not use Descriptor Trap 20256 or Command Trap 27018 as a source or destination, may not load or invalidate registers in ATU 10228 or Protection Cache 10234, and may not perform physical memory references, i.e., memory references which place physical descriptors directly on PD Bus 10146, instead of presenting logical

descriptors to Memory Reference Unit 27017 and Protection Unit 27019. Similarly, non-KOS microcode may not specify KOS registers in the GRs 10360 of GRF 10354 or the bottom frame of the stack portion of GRF 10354 when addressing GRF 10354 registers. Further, in embodiments allowing dynamic loading of FUSITT 11012, only KOS microcode may manipulate the devices provided for dynamic loading.

In a present embodiment of CS 10110, the distinction between KOS devices and registers and devices and registers accessible to all microprograms is maintained by the microbinder. The microbinder checks all microcode for microinstructions which manipulate devices in Protection Unit 27019, or Memory Reference Unit 27017, or which address GRF 10354 registers reserved for KOS use. However, it is characteristic of the micromachine that KOS devices are logically and physically separate from devices accessible to all microprograms and, consequently, other embodiments of CS 10110 may use hardware devices to prevent non-KOS microprograms from manipulating KOS devices.

c. Micromachine Stacks and Microroutine Calls and Returns (Figs. 272, 273)

1. Micromachine Stacks (Fig. 272)

As previously mentioned, the FU micromachine is a stack micromachine. The properties of the FU micromachine's stack depends on whether the FU micromachine is in virtual or monitor mode. In virtual mode, the micromachine stack is of essentially unlimited size; if it contains more frames than allowed for inside FU 10120, the top frames are in GRF 10354 and the remaining frames are in Secure Stack 10336 belonging to Process 610 being executed by the FU micromachine. In the following, the virtual mode micromachine stack is termed the virtual micromachine stack. In monitor mode, the micromachine stack consists of a fixed amount of storage; in a present embodiment of CS 10110, the monitor mode micromachine stack is completely contained in the stack portion, SRs 10362, of GRF 10354; in other embodiments of CS 10110, part or all of the monitor mode micromachine stack may be contained in an area of MEM 10112 which has a fixed size and a fixed location known to the monitor micromachine. In yet other embodiments of CS 10110, monitor mode micromachine stack may be of flexible depth in a manner similar to the virtual micromachine stack. In either mode, microroutines other than certain KOS microroutines which execute state save and restore operations may access only two frames of GRF 10354 stack: the frame upon which the microroutine is executing, called the current frame, and the frame upon which the microroutine that invoked that microroutine executed, called the previous frame. KOS microroutines which execute state save and restore operations may in addition access the bottom frame of that portion of the virtual micromachine stack which is contained in GRF 10354.

Fig. 272 illustrates stacks for the FU micromachine. Those portions of the micromachine stack which are contained in the FU are contained in SR's 10362 (of GRF 10354) and in RCWS 10358. Each register of RCWS 10358 is permanently associated with a GRF frame in SRs 10362 of GRF 10354, and the RCWS 10358 register and the GRF frame together may contain one frame of a micromachine stack. As previously describe, each register of GRF 10354 contains three fields: one for an AON and other information, one for an offset, and one for a length. As illustrated in Fig. 251, each register in RCWS 10358 contains four fields:

- A one bit field which retains the value of the Condition Code register in MCW1 20290 at the time that the invocation which created the next frame occurred.
- A field indicating what Event Signals were pending at the time that the invocation to which the RCWS register belongs invoked another microroutine.
- A flag indicating whether the microinstruction being executed when the invocation occurred was the first microinstruction in an SIN.
- The address at which the execution of the invoking microroutine is to continue.

The uses of these fields will become apparent in the ensuing discussion.

The space available for micromachine stacks in SRs 10362 and RCWS 10358 is divided into two parts: Frames 27205 reserved for MOS 10370 and Frames 27206 available for the MIS 27203. Frames 27206 may contain no MIS Frames 27203, or be partially or completely occupied by MIS Frames 27203. Space which contains no MIS Frames is Free Frames 27207. The size of the space reserved for Monitor Micromachine Stack Frames 27205 is fixed, and Spaces 27203, 27205, and 27207 always come in the specified order. Register Addressing 27011 handles addressing in Stack Portion 27201 of GRF 10354 and RCWS 10358 in such fashion that the values for the locations of current, previous, and bottom frames specifying registers in RCWS 10358 or frames in Stack Portion 27201 automatically "wrap around" when they are incremented beyond the largest index value allowed by the sizes of the registers or decremented below the smallest index value. Thus, though Spaces 27203, 27205, and 27207 always have the same relative order, their GRF 10354 frames and RCWS registers may be located anywhere in Stack Portion 27201 and RCWS 10358.

2. Microroutine Invocations and Returns

In CS 10110, microroutines may be invoked by other microroutines or by signals from CS 10110 hardware. The methods of invocation aside, microroutine invocations and returns resemble invocations of and returns from procedures written in high-level languages. In the following, the general principles of microroutine invocations and returns are discussed, and thereafter, the specific methods by which microroutines may be invoked in CS 10110. The differences between invocations in monitor mode and

invocations in virtual mode are explained in the detailed discussions of the two modes.

The microroutine which is currently being executed runs on the frame specified by Current Pointer 27215. When an invocation occurs, either because the executing microroutine performs a call, or because a signal which causes invocations has occurred, JP 10114 hardware does three things:

- 5 — It stores state information for the invoking microroutine in the RCWS 10358 register associated with the current frame. The state information includes the location at which execution of the invoking microroutine will resume, as well as other state information.
- It increments Current Pointer 27215 and Previous Pointer 27213, thereby providing a frame for the new invocation.
- 10 — It begins executing the first instruction of the newly invoked microroutine.

Because the newly-invoked microroutine can access registers of the invoking microroutine's frame, the invoking microroutine can pass "arguments" to the invoked microroutine by placing values in registers in its frame used by the invoked microroutine. However, the invoking microroutine cannot specify which registers contain "arguments" on an invocation, so the invoked microroutine must know which registers of the previous frame are used by the invoking microroutine. Since the only "arguments" which a microroutine has access to are those in the previous frame, a microroutine can pass arguments which it received from its invoker to a microroutine which it invokes only by copying the arguments from its invoker's frame to its own frame; which then becomes the newly-invoked routine's previous frame.

The return is the reverse of the above: Current Pointer 27215 and Previous Pointer 27213 are decremented, thereby "popping off" the finished invocation's frame and returning to the invoker's frame. The invoker then resumes execution at the location specified in the RCWS 10358 register and using the state saved in the RCWS 10358. The saved state includes the value of the Condition Code in MCW1 20290 at the time of the invocation and flags indicating various pending Events. The Condition Code field in MCW1 20290 is set to the saved value, and the pending event flags may cause Events to occur as described in detail below.

3. Means of Invoking Microroutines

In the micromachine, invocations may be produced either by commands in microinstructions or by hardware signals. In the following, invocations produced by commands in microinstructions are termed Calls, while those produced by hardware signals are termed Event invocations and Jams. Invocations are further distinguished from each other by the locations to which they return. Calls and Jams return to the microinstruction following the microinstruction in which the invocation occurs; Event invocations return to that microinstruction, which is then repeated.

In terms of implementation, the different return locations are a consequence of the point in the micromachine cycle at which Calls, Jams, and Event invocations save a return location and transfer control to the called routine. With Calls and Jams, these operations are performed in the M1 cycle; with Event invocations, on the other hand, the Event signal during the M0 cycle causes the M0 cycle to be followed by a MA cycle instead of the M1 cycle, and the operations are performed in the MA cycle. In the M1 cycle, the value in mPC 20276 is incremented; in the MA cycle, it is not. Consequently, the return value saved in RCWS 10358 on a Call or Jam is the incremented value of mPC 20276, while the return value saved on an Event invocation is the unincremented value of mPC 20276. The following discussion will deal first with Calls and Jams, and then with Event invocations.

A Call command in a microinstruction contains a literal value which specifies the offset from the microinstruction containing the Call at which execution is to continue after the Call. When the microinstruction with the Call command is executed in micromachine cycle M1, BRCASE 20278 adds the offset contained in the command to the current value of mPC 20276 in order to obtain the location of the invoked microroutine and sets SITTNAS 20286 to select the location provided by BRCASE 20278 as the location of the next microinstruction. Then the Call command increments mPC 20276 and stores the incremented value of mPC 20276 in the RCWS 10358 register associated with the current frame in SRs 10362 and increments Current Pointer 27215 and Previous Pointer 27213 to provide a new frame in SRs 10362. The Jam works exactly like the Call, except that a hardware signal during micromachine cycle M1 causes the actions associated with the invocation to occur and provides the location of the invoked microroutine directly to SITTNAS 20286.

With Events, Event Logic 20284 causes an invocation to occur during cycle M0 and provides the location of the invoked microroutine via CSADR 20299. Since the Event occurs during cycle M0, the location stored in RCWS 10358 is the unincremented value of mPC 20276, and SITTNAS 20286 selects the location provided by Event Logic 20284 as the location of the next microinstruction. Since the return from the Event causes the microinstruction during which the Event occurred to be re-executed, the microroutine and the microroutine to which it belongs may be said to be "unaware" of the Event's occurrence. The only difference between the execution of a microinstruction during which an Event occurs and the execution of the same microinstruction without the Event is the length of time required for the execution.

4. Occurrence of Event Invocations (Fig. 273)

As described previously, Event invocations are produced by Event Logic 20284. The location in microcode to which Event Logic 20284 transfers control is determined by the following:

- The operation being commenced by FU 10120. Certain Event invocations may occur only at the beginning of certain FU 10120 operations.
- The state of Event signal lines from hardware and internal registers in Event Logic 20284.
- The state of certain registers visible via MCW1 20290. Some of these registers enable Events and others mask Events. Of the registers which enable Events, some are set by Event signals and others by the microprogram.
- On returns from invocations of microroutines, the settings of certain bits in the RCWS 10358 register belonging to the micromachine frame for the invocation that is being returned to.

Microprograms may use these mechanisms to disable Event signals and to delay an Event Invocation from an Event signal for a single microinstruction or an indefinite period, and FU 10120 uses them to automatically delay Event invocations resulting from certain Event signals. Using traditional programming terminology, the mechanisms allow a differential masking of Event signals. An Event signal may be explicitly masked for a single microinstruction, it may be masked for a sequence of microinstructions; it may be automatically masked until a certain operation occurs, or it may be automatically masked for a certain maximum length of time. Event signals which occur while they are masked are not lost. In some cases, the Event signal continues until it is serviced; in others, a register is set to retain the fact that the Event signal occurred. When the Event signal is unmasked, the set register causes the Event signal to reoccur. In some cases, finally, the Event signal is not retained, but recurs when the microinstruction which caused it is repeated.

In the following, the relationship between FU 10120 operations and Event signals is first presented, and then a detailed discussion of the enabling registers in MCW1 20290 and of the bits in RCWS 10358 registers which control Event invocations.

FU 10120 allows Event invocations resulting from Event signals to be inhibited for a single microinstruction; it also delays certain Event invocations for certain Event signals until the first microinstruction of an SIN. Other Event signals occur only at the beginning of an SIN, at the beginning of a Namespace Resolve or Evaluate operation, or at the beginning of a logical memory reference.

Event invocations may be delayed for a single microinstruction by setting a field of the microinstruction itself. Setting this field delays almost all Event invocations, and thereby guarantees that an Event invocation will not occur during the microinstruction's M0 cycle.

Event signals relating to debugging occur at the beginnings of certain micromachine operations. Such Event signals are called Trace Event signals. As will be explained in detail, in the discussion of the debugger, Trace Event signals can occur on the first microinstruction of an SIN, at the beginning of an Evaluate or Resolve operation, at the beginning of a logical memory reference, or at the beginning of a microinstruction. IPM interrupt signals and Interval Timer Overflow Event signals are automatically masked until the beginning of the next SIN or until a maximum amount of time has elapsed, whichever occurs first. The mechanisms involved here are explained in detail in the discussion of interrupt handling in the FU 10120 micromachine.

Turning now to the registers used to mask and enable Event signals, Fig. 273 is a representation of the masking and enabling registers in MCW1 20290 and of the field in RCWS 10358 registers which controls Event invocations. Beginning with the registers in MCW1 20290, there are three registers which control Event invocations: Event Mask Register (EM) 27301, Events Pending Register (EP) 27309, and Trace Enable Register (TE) 27319. Bits in EM 27301 mask certain Event signals as long as they are set; bits in EP Register 27309 record the occurrence of certain Event signals while they are masked; when bits in TE Register 27319 are set, Trace Event signals occur before certain FU 10120 operations.

EM 27301 contains three one bit fields: Asynchronous Mask Field 27303, Monitor Mask Field 27305, and Trace Event Mask Field 27307. As explained in detail in the discussion of FU 10120 hardware, these bits establish a hierarchy of Event masks. If Asynchronous Mask Field 27303 is set, only two Event signals are masked: that resulting from an overflow of EGGTMR 25412 and that resulting from an overflow of EU 10122's stack. If Monitor Mask Field 27305 is set, those Events are masked, and additionally, the FU Stack Overflow Event signal is masked. As will be explained in detail later, when the FU 10120 Stack Overflow Event signal is masked, the FU micromachine is executing in monitor mode. If Trace Event Mask Field 27307 is set, Trace Trap Event signals are masked in addition to the above signals. Each of the fields in EM 27301 may be individually set and cleared by the microprogram.

Four Event signals set fields in EP 27309: the EGGTMR 25412 Runout signal sets ET Field 27311, the INTTMR 25410 Runout signal sets IT Field 27313, the Non-Fatal Memory Error signal sets ME Field 27315, and the Inter-Process Message signal sets IPM Field 27317. Event invocations for all of these Event signals but the Egg Timer Runout signal occur at the beginning of an SIN; in these cases the fields in EP 27309 retain the fact that the Event signal has occurred until that time; the Event invocation for the Egg Timer Runout signal occurs as soon after the signal as the settings of mask bits in EM 27301 allow. The bit in ET Field 27311 retains the fact of the Egg Timer Runout signal until the masking allows the Event invocation to occur. All of the fields in EP 27309 but ME Field 27315 may be reset by microcode. The microroutines invoked by the Events must reset the appropriate fields; otherwise, they will be reinvoked when they

return. ME Field 27315 is automatically reset when the memory error is serviced.

TE Register Field 27319 enables tracing. Each bit in the register enables a kind of Trace Event signal when it is set. Depending on the kind of tracing, the Trace Event signal occurs at the beginning of an SIN, at the beginning of a Resolve or Evaluate operation, at the beginning of a logical memory reference, or at the beginning of a microinstruction. For details, see the following description of debugging.

Turning now to the registers contained in RCWS 10358, each RCWS Register 27322 contains eight fields which control Event signals. The first field is FM Field 27323. FM Field 27323 reflects the value of a register in Event Logic 20284 when the invocation to which RCWS Register 27322 belongs occurs. The register in Event Logic 20284 is set only when the microinstruction currently being executed is the first microinstruction of an SIN. Thus, FM Field 27323 is set only in RCWS Registers 27322 belonging to Event invocations which occur in the M0 cycle of the first microinstruction in the SIN, i.e., at the beginning of the SIN. The value of the register in Event Logic 20284 is saved in FM Field 27323 because several Event invocations may occur at the beginning of a single SIN. The Event invocations occur in order of priority: when the one with the highest priority returns, the fact that FM Field 27323 is set causes the register in Event Logic 20284 to again be set to the state which it has on the first microinstruction of an SIN. The register's state, thus set, causes the next Event invocation which must occur at the beginning of the SIN to take place. After all such invocations are finished, the first microinstruction enters its M1 cycle and resets the register in Event Logic 20284. In its reset state, the register inhibits all Event invocations which may occur only at the beginning of an SIN. It is again set at the beginning of the next SIN.

The remaining fields in RCWS Register 27322 which control Event invocations are the fields in Return Signals Field 27331. These fields allow the information that an Event signal has occurred to be retained through Event invocations until the Event signal's Event invocation takes place. When an invocation occurs, these fields are set by Event Logic 20284. On return from the invocation, the values of the fields are input into Event Logic 20284, thereby producing Event signals. The Event signal with the highest priority results in an Event invocation, and the remaining Event signals set fields in Return Signals Field 27331 belonging to RCWS Register 27322 belonging to the invocation which is being executed when the Event signals occur. Because the fields in Return Signals Field 27330 are input into Event Logic 20284, microcode invoked as a consequence of Event signals which sets one of these fields must reset the field itself. Otherwise, the return from the microcode will simply result in a reinvocation of the microcode.

The seven fields in Return Signals Field 27330 have the following significance:

- When EG Field 27333 is set, an EU 10122 dispatch operation produced an illegal location in EU 10122 microcode EUSITT 20344.
- When NT Field 27335, ST Field 27341, mT Field 27343, or mB Field 27345 is set, a trace signal has occurred. These are explained in detail in the discussion of debugging.
- When ES Field 27337 is set, an EU 10122 Storeback Exception has occurred, i.e., an error occurred when EU 10122 attempted to store the result of an operation in MEM 10112.
- When MRR Field 27339 is set, a condition such as an ATU 10228 miss or a Protection Cache 10234 miss has occurred, and it is necessary to reattempt a memory reference.

d. Virtual Micromachines and the Monitor Micromachine

As previously described, microcode being executed on FU 10120's micromachine can run in either monitor mode or virtual mode. In this portion of the discussion, the distinguishing features and applications of the two modes are explained in detail.

1. Virtual Mode

As previously mentioned, the chief distinction between virtual mode and monitor mode is MIS 10368. The fact that MIS 10368 is of essentially unlimited size has the following consequences for microroutines which execute in virtual mode.

- An invocation of a microroutine executing in virtual mode may have as its consequence further invocations to any depth.
- Any invocation of or return from a microroutine executing in virtual mode may cause a page fault. The FU micromachine is in virtual mode when all bits in the Event Masks portion of MCW1 20290 are cleared. In this state, no enabled Event signals are masked, and Event invocations may occur in any microinstruction which does not itself mask them.

Because invocations may occur to any depth in virtual mode, microroutines executing in this mode may be recursive. Such recursive microroutines are especially useful for the interpretation of Names. Often, as previously described, the Name Table Entry for a Name will contain Names which resolve to other Names, and the virtual micromachine's limitless stack allows the use of recursive Name Resolution microroutines in such situations. Recursive microroutines may also be used for complex SINs, such as Calls.

Because invocations can occur to any depth, any number of Events may occur while a microroutine is executing in monitor mode. This in turn greatly simplifies Event handling. If an Event signal occurs while an Event with a given priority is being handled and the Event being signalled has a higher priority than the one

being handled, the result is simply the invocation of the new Event's handler. Thus, the order in which the Event handlers finish corresponds exactly to the priorities of their Events: those with the highest finish first.

A page fault may occur on any microinvocation or return executed in virtual mode because an invocation in virtual mode which occurs when there are no more Free Frames 27207 on SRs 10362 causes an Event signal which invokes a microroutine running in monitor mode. The microroutine transfers MIS Frames 27203 from GRF 10354 to Secure Stack 10336 in MEM 10112, and the transfer may cause a page fault. Similarly, when a microreturn takes place from the last frame on MIS Frames 27203 on SRs 10362, an Event signal occurs which invokes a microroutine that transfers additional frames from Secure Stack 10336 to GRF 10354, and this transfer, too, may cause a page fault.

The fact that page faults may occur on microinvocations or microreturns in virtual mode has two important consequences: microroutines which cannot tolerate page faults other than those explicitly generated by the microroutine itself cannot execute in virtual mode, and because unexpected page faults cause execution to become indeterminate, microroutines which must run to completion cannot execute in virtual mode. For example, if the microroutine which handles page faults executed in virtual mode, its invocation could cause a page fault, which would cause the microroutine to be invoked again, which would cause another page fault, and so on through an infinite series of recursions.

2. Monitor Micromachine

As previously described, the essential feature of monitor mode is MOS 10370. In a present embodiment of CS 10110, this stack has a fixed minimum size, and is always contained in GRF Registers 10354. The nature of MOS 10370 has four consequences for microroutines which execute in monitor mode:

- When the micromachine is in monitor mode, the depth of invocations is limited; recursive microroutines therefore cannot be executed in monitor mode, and Event invocations must be limited.
- Invocations of microroutines or returns from microroutines in monitor mode never result in page faults.
- Microroutines executing in monitor mode are guaranteed to run to completion if they do not suspend the Process 610 which they are executing or perform a Call to software.
- When the micromachine is executing in monitor mode, it is guaranteed to return to virtual mode within a reasonable period of time, either because a microroutine executing in monitor mode has run to completion, or because the microroutine has suspended the Process 610 which it is executing, or has made a Call to software. The result in both cases is the execution of a new sequence of SOPs, and thus a return to virtual mode.

In a present embodiment of CS 10110, the FU micromachine is in monitor mode when a combination of masking bits in MCW1 20290 is set which results in the masking of the FU Stack Overflow Event and the Egg Timer Overflow Event. As previously described, these Events are masked if Fields 27303, 27305, or 27307 is set. These Events and the consequences of masking them are explained in detail below.

The event signal for the FU Stack Overflow Event occurs on microinvocations for which there is no frame available in MIS Frames 27203. If the Event signal is not masked, it causes the invocation of a microroutine which moves MIS Frames from MIS Frames 27203 onto a Process 610's Secure Stack 10336. When the FU Stack Overflow Event is masked, all frames in SRs 10362 of GRs 10360 are available for microroutine invocations and microroutine invocations will not result in page faults, but if the capacity of SRs 10362 is exceeded, FU 10120 ceases operation.

The Egg Timer Overflow event signal occurs when Egg TMR 25412 runs out. As will be explained in detail later, Egg TMR 25412 ensures that an Interval Timer Runout, an Inter-processor Message, or a Non-fatal Memory Error will be serviced by JP 10114 within a reasonable amount of time. If an Interval Timer Runout Event signal or an Inter-processor Message Event signal occurs at a time when it is inefficient for the FU micromachine to handle the Event, Egg TMR 25412 begins running. When Egg TMR 25412 runs out, the Event is handled unless the micromachine is in monitor mode. If the Egg TMR 25412 Runout Event signal occurs while the FU micromachine is in monitor mode, i.e., while the Event is masked, the Event signal sets Field 27311 in MCW1 20290. When the FU micromachine reverts to virtual mode, i.e., when all Event Mask bits in MCW1 20290 are cleared, the Egg TMR 25412 Runout Event occurs, and the Interval Timer Runout Inter-processor Message Event handlers are invoked by Event Logic 20284.

e. Interrupt and Fault Handling

1. General Principles

Any computer system must be able to deal with occurrences which disrupt the normal execution of a program. Such occurrences are generally divided into two classes: faults and interrupts. A fault occurs as a consequence of an attempt to execute a machine instruction, and its occurrence is therefore synchronous with the machine instruction. Typical faults are floating point overflow faults and page faults. A floating point overflow fault occurs when a machine instruction attempts to perform a floating point arithmetic operation and the result exceeds the capacity of the CS 10110's floating point hardware, that is EU 10122. A page fault occurs when a machine instruction in a computer system with virtual memory attempts to reference data which is not presently available in the computer system's primary memory, that is MEM

10112. Since faults are synchronous with the execution of machine instructions and in many cases the result of the execution of specific machine instructions, their occurrence is to some extent predictable.

The occurrence of an interrupt is not predictable. An interrupt occurs as a consequence of some action taken by the computer system which has no direct connection with the execution of a machine instruction by the computer system. For example, an I/O interrupt occurs when data transmitted by an I/O device (IOS 10116) reaches the central processing unit (FU 10120), regardless of the machine instruction the central processing unit is currently executing.

In conventional systems, interrupts and faults have been handled as follows: if an interrupt or fault occurs, the computer system recognizes the occurrence before it executes the next machine instruction and executes an interrupt-handling microroutine or Procedure 602 instead of the next machine instruction. If the interrupt or fault cannot be handled by the Process 610 in which it occurs, the interrupt or fault results in a process swap. When the interrupt handling routine is finished, Process 610 which faulted or was interrupted can be returned to the CPU if it was removed and the next machine instruction executed.

While the above method works well with faults, the fact that interrupts are asynchronous causes several problems:

— Machine instructions cannot require an indefinite amount of time to execute, since interrupts cannot be handled until the machine instruction during which they occur is finished.

— It must be possible to remove a Process 610 from the CPU at any time, since the occurrence of an interrupt is not predictable. This requirement greatly increases the difficulty of process management.

The method used for interrupt and fault handling in a present embodiment of CS 10110 is described below.

2. Hardware Interrupt and Fault Handling in CS 10110

In CS 10110, there are two levels of interrupts: those which may be created and dealt with completely by software, and those which may be created by hardware signals. The former class of interrupts is dealt with in the discussion of Processes 610; the latter, termed hardware interrupts, is discussed below.

In CS 10110, hardware interrupts and faults begin as invocations of microroutines in FU 10120. The invocations may be the result of Event signals or may be made by microprograms. For example, when IOS 10116 places data in MEM 10112 for JP 10114, an Inter-processor Message Event signal results, and the signal causes the invocation of Inter-processor Message Interrupt handler microcode. On the other hand, a Page Fault begins as an invocation of Page Fault microcode by LAT microcode. The actions taken by the microcode which begins handling the fault or interrupt depend on whether the fault or interrupt is handled by the Process 610 which was being executed when the fault or Event occurred or by a special KOS Process 610.

In the first case, the Event microcode may perform a Microcode-to-Software Call to a high-level language procedure which handles the Event. An example of an Event handled in this fashion is a floating point overflow: when FU 10120 microcode determines that a floating point overflow has occurred, it invokes microcode which may invoke a floating point overflow procedure provided by the high-level language whose S-Language was being executed when the overflow occurred. In alternate embodiments of CS 10110, the overflow procedure may also be in microcode.

In the second case, the microcode handling the fault or interrupt puts information in tables used by a KOS Process 610 which handles the fault or interrupt and then causes the KOS Process 610 to run at some later time by advancing an Event Counter awaited by the Process 610. Event Counters and the operations on them are explained in detail in a following description of Processes 610. Since the tables and Event Counters manipulated by microcode are always present in MEM 10112, these operations do not cause page faults, and can be performed in monitor mode. For example, when IOS 10116 transmits an IPM Event signal to JP 10114 after IOS 10116 has loaded data into MEM 10112, the Event resulting from the Event signal invokes microcode which examines a queue containing messages from IOS 10116. The messages in the queue contain Event Counter locations, and the microcode which examines the queue advances those Event counters, thereby causing Processes 610 which were waiting for the data returned by the I/O operation to recommence execution.

3. The Monitor Mode, Differential Masking and Hardware Interrupt Handling

FU 10120 micromachine's monitor mode and differential masking facilities allow a method of hardware interrupt handling which overcomes two problems associated with conventional hardware interrupt handling: an interrupt can be handled in a predictable amount of time regardless of the amount of time required to execute an SIN, and if the microcode which handles the interrupt executes in monitor mode, the interrupt may be handled at any time without unpredictable consequences. There are two sources of hardware interrupts in CS 10110: an Inter-Processor Message (IPM) and an Interval Timer 25410 Runout. An IPM occurs when IOS 10116 completes an I/O task for JP 10114 and signals completion of the task via IQJP Bus 10132. An Interval Timer Runout occurs when a preset time at which CS 10110 must take some action is reached. For example, a given Process 610 may have a limit placed on the amount of time it may execute on JP 10114. As is explained in a following description of process synchronization, the virtual processor management system sets Interval Timer 25412 to run out when Process 610 has used all of the time available to it.

Both IPMs and Interval Timer Runouts begin as Event signals. The immediate effect of the Event signal is to set a bit in EP Field 27309 of MCW1. In principle, the set bit can cause invocation of the event microcode for the Event on the next M0 cycle in which the FU 10120 micromachine is in virtual mode. Since microroutines running in monitor mode are guaranteed to return the micromachine to virtual mode within a reasonable length of time, and the Event invocation will occur when this happens, the Event is guaranteed to be serviced in a reasonable period of time. The microroutines invoked by the Events themselves execute in monitor mode, thereby guaranteeing that no page faults will occur while they are executing and that Process 610 which is executing on JP 10114 when the hardware interrupt occurs need not be removed from JP 10114.

While hardware interrupts are serviced in principle as described above, considerations of efficiency require that as many hardware interrupts as possible be serviced when the size of the FU micromachine's stack is at a minimum, i.e., at the beginning of an SIN's execution. This requirement is achieved by means of Egg TMR 25412 and ET Flag 27311 in MCW1 20290. As described above, when an IPM interrupt or an Interval Timer 25410 Runout interrupt occurs, Field 27317 or 27313 respectively is set in MCW1 20290. At the same time, Egg TMR 25412 begins running. If the current SIN's execution ends before Egg TMR 25412 runs out, the set Field in MCW1 20290 causes the Interval Timer Runout or Inter-processor Message Event invocations to occur on the first microinstruction for the next SIN. If, on the other hand, the current SIN's execution does not end before Egg TMR 25412 runs out, the Egg Timer Runout causes an Event signal. The immediate result of this signal is the setting of ET bit 27311 in MCW1 20290, and the setting of ET bit 27311 in turn causes the Interval Timer Runout Event invocation and/or IPM Event invocation to take place on the next M0 cycle to occur while the micromachine is in virtual mode. The above mechanism thus guarantees that most hardware interrupts will be handled at the beginning of an SIN, but that hardware interrupts will always be handled within a certain amount of time regardless of the length of time required to execute an SIN.

g. FU Micromachine and CS 10110 Subsystems

The subsystems of CS 10110, such as the object subsystem, the process subsystem, the S-Interpreter subsystem, and the Name Interpreter subsystem, are implemented all or in part in the micromachine. The description of the micromachine therefore closes with an overview of the relationship between these subsystems and the micromachine. Detailed descriptions of the operation of the subsystems have been presented previously.

The subsystems fall into three main groups: KOS subsystems, the Name Interpreter subsystem, and the S-Interpreter subsystem. The relationship between the three is to some extent hierarchical: the KOS subsystems provide the environment required by the Name Interpreter subsystem, and the Name Interpreter subsystem provides the environment required by the S-Interpreter subsystem. For example, the S-Interpreter subsystem interprets SINs consisting of SOPs and Names; the Name Interpreter subsystem translates Names into logical descriptors, using values called ABPs to calculate the locations contained in the logical descriptors. The KOS subsystems calculate the values of the ABPs, translate Logical Descriptors 27116 into physical MEM 10112 addresses, and check whether a Process 610 has access to an object which it is referencing.

In a present embodiment of CS 10110, the Name Interpreter subsystem and the S-Interpreter subsystem are implemented completely in the micromachine; in other embodiments, they could be implemented in high-level languages or in hardware. The KOS subsystems are implemented in both the micromachine and in high-level language routines. In alternate embodiments of CS 10110, KOS subsystems may be embodied entirely in microcode, or in high-level language routines. Some high-level language routines may execute in any Process 610, while others are executed only by special KOS Processes 610. The KOS subsystems also differ from the others in the manner in which the user has access: with the S-Interpreter subsystem and the Name Interpreter subsystem, the subsystems come into play only when SINs are executed; the subsystems are not directly visible to users of the system. Portions of the KOS subsystems, on the other hand, may be explicitly invoked in high-level language programs. For example, an invocation in a high-level language program may cause KOS to bind a Process 610 to a Virtual Processor 612.

The following will first list the functions performed by the subsystems, and then relate the subsystems to the monitor and virtual micromachine modes and specific micromachine devices. KOS subsystems perform the following functions:

- Virtual memory management;
- Virtual processor management;
- Inter-processor communication;
- Access Control;
- Object management; and,
- Process management.

The Name Interpreter performs the following functions:

- Fetching and parsing SOPs, and
- Interpreting Names.

The S-Interpreter, finally, dispatches SOPs, i.e., locates the FU 10120 and EU 10122 microcode which

executes the operation corresponding to a given SOP for a given S-Language.

Of these subsystems, the S-Interpreter, the Name Interpreter, and the microcode components of the KOS process and object manager subsystems execute on the virtual micromachine; the microcode components of the remaining KOS subsystems execute on the monitor micromachine. As will be seen in the discussions of these subsystems, subsystems which execute on the virtual micromachine may cause Page Faults, and may therefore reference data located anywhere in memory; subsystems which execute on the monitor micromachine may not cause Page Faults, and the data bases which these subsystems manipulate must therefore always be present at known locations in MEM 10112.

The relationship between subsystems and FU 10120 micromachine devices is the following: Microcode for all subsystems uses DESP 20210, Microcode Addressing 27013, and Register Addressing 27011, and may use EU Interface 27007. S-Interpreter microcode uses SOP Decoder 27003, and Name Interpreter Microcode uses Instruction Stream Reader 27001, Parsing Unit 27005, and Name Translation Unit 27015. KOS virtual memory management microcode uses Memory Reference Unit 27017, and Protection Microcode uses Protection Unit 27019.

Having described in detail the structure and operation of CS 10110's major subsystems, MEM 10112, FU 10120, EU 10122, IOS 10116, and DP 10118, and the CS 10110 micromachine, CS 10110 operation will be described in further detail next below. First, operation of CS 10110's Namespace, S-Interpreter, and Pointer Systems will be described. Then, operation of CS 10110 will be described in further detail with respect to CS 10110's Kernel Operating System.

3. Namespace, S-Interpreters, and Pointers (Figs. 301—307, 274)

The preceding chapters have presented an overview of CS 10110, examined its hardware in detail, and explained how the FU 10120 hardware functions as a micromachine which controls the activities of other CS 10110 components. In the remaining portions of the specification, the means are presented by which certain key features of CS 10110 are implemented using the hardware, the micromachine, tables in memory, and high-level language programs. The present chapter presents three of these features: the Pointer Resolution System, Namespace, and the S-Interpreters.

The Pointer Resolution System translates pointers, i.e., data items which contain location information, into UID-offset addresses. Namespace has three main functions:

- It locates SInS and fetches them from CS 10110's memory into FU 10120.
- It parses SInS into SOPs and Names.
- It translates Names into Logical Descriptors 27116 or values.

The S-Interpreters decode S-operations received from namespace into locations in microcode contained in FUSITT 11012 and EUSITT 20344 and then execute that microcode. If the S-operations require operands, the S-Interpreters use Namespace to translate the operands into Logical Descriptors 27116 or values as required by the operations.

Since Namespace depends on the Pointer Resolution System and the S-Interpreters depend on Namespace, the discussion of the systems begins with pointers and then deals with namespace and S-Interpreters.

A. Pointers and Pointer Resolution (Figs. 301, 302)

A pointer is a data item which represents an address, i.e., in CS 10110, a UID-offset address. CS 10110 has two large classes of pointers: resolved pointers and unresolved pointers. Resolved pointers are pointers whose values may be immediately interpreted as UID-offset addresses; unresolved pointers are pointers whose values must be interpreted by high level language routines or microcode routines to yield UID-offset addresses. The act of interpreting an unresolved pointer is called resolving it. Since the manner in which an unresolved pointer is resolved may be determined by a high-level language routine written by a system user, unresolved pointers provide a means by which users of the system may define their own pointer types.

Both resolved and unresolved pointers have subclasses. The subclasses of resolved pointers are UID pointers and object relative pointers. UID pointers contain a UID and offset, and can thus represent any CS 10110 address; object-relative pointers contain only an offset; the address's UID is assumed to be the same as that of the object containing the object-relative pointer. An object-relative pointer can therefore only represent addresses in the object which contains the pointer.

The subclasses of unresolved pointers are ordinary unresolved pointers and associative pointers. The difference between the two kinds of unresolved pointers is the manner in which they are resolved. Ordinary unresolved pointers are always resolved by high-level language routines, while associative pointers are resolved the first time they are used in a Process 610 and a domain by high-level language routines, but are subsequently resolved by means of a table called the Associated Address Table (AAT). This table is accessible to microcode, and associative pointers may therefore be more quickly resolved than ordinary unresolved pointers.

The following discussion will first explain the formats used by all CS 10110 pointers, and will then explain how pointers are processed in FU 10120.

a. Pointer Formats (Fig. 301)

Figure 301 represents a CS 10110 pointer. The figure has two parts: a representation of General Pointer Format 30101, which gives an overview of the fields which appear in all CS 10110 pointers, and a detailed presentation of Flags and Format Field 30105, which contains the information by which the kinds of CS 10110 pointers are distinguished.

Turning first to General Pointer Format 30101, all CS 10110 pointers contain 128 bits and are divided into three main fields:

- Offset Field 30103 contains the offset portion of a UID-offset address in resolved pointers and in associative pointers; in other unresolved pointers, it may contain an offset from some point in an object or other information as defined by the user.

- Flags and Format Field 30105 contains flags and format codes which distinguish between kinds of pointers. These flags and format codes are explained in detail below.

- UID field 30115 contains a UID in UID pointers and in some associative pointers; in objectrelative pointers, and other associative pointers, its meaning is undefined, and in ordinary unresolved pointers, it may contain information as defined by the user.

Flags and Format Field 30105 contains four subfields:

- Fields 30107 and 30111 are reserved and must be set to 0.

- NR Field 30109 indicates whether a pointer is resolved or unresolved. In resolved pointers, the field is set to 0, and in unresolved pointers, it is set to 1.

- Format Code Field 30113 indicates the kind of resolved or unresolved pointers. Format codes for the present embodiment are explained below.

The values of Format Code Field 30113 may range from 0 to 31. If Format Code Field 30113 has the value 0, the pointer is a null pointer, i.e., a pointer which neither directly nor indirectly indicates an address. The meanings of the other format codes depend on the value of NR Field 30109:

NR Field Value	Format Code Value	Meaning
0	1	UID pointer
0	2	Object-relative pointer
0	all other codes	Illegal
1	1	UID associative pointer
1	2	Object-relative associative pointer
1	all other codes	Ordinary unresolved pointer

As indicated by the above table, the present embodiment has two kinds of associative pointer, UID associative pointers and object-relative associative pointers. Like a UID pointer, a UID associative pointer contains a UID and an offset, and like an object-relative pointer, an object-relative associative pointer contains an offset and takes the value of the UID from the object to which it belongs. However, as will be explained in detail later, the UID and offset which the associative pointers contain or represent are not used as addresses. Instead, the UID and offset are used as tags to locate entries in the AAT, which associates an associative pointer with a resolved pointer.

b. Pointers in FU 10120 (Fig. 302)

When a pointer is used as an address in FU 10120, the address information in the pointer must be translated into a Logical Descriptor 27116 consisting of an AON, an offset, and a length field of 0; when a Logical Descriptor 27116 in FU 10120 is used to form a pointer value in memory, the AON must be converted back to a UID. The first conversion is termed pointer-to-descriptor conversion, and the second descriptor-to-pointer conversion. Both conversions are accomplished by microcodes executing in FU 10120.

What is involved in the translation depends on the kind of pointer: if the pointer is a UID pointer, the UID must be translated into an AON; if the pointer is an object-relative pointer, the AON required to fetch the pointer is the pointer's AON, so no translation is necessary. If the pointer is an unresolved pointer, it must first be translated into a resolved pointer and then into a Logical Descriptor 27116. If the pointer is associative, the translation to a resolved pointer may be performed by means of the ATT.

In the present embodiment, when other FU 10120 microcode calls pointer-to-descriptor microcode, the calling microcode passes Logical Descriptor 27116 for the location of the pointer which is to be translated as an argument to the pointer-to-description translation microcode. The pointer-to-descriptor microcode returns a Logical Descriptor 27116 produced from the value of the pointer at the location specified by

Logical Descriptor 27116 which the pointer-to-descriptor microcode received as an argument.

The pointer-to-descriptor microcode first uses Logical Descriptor 27116 given it as an argument to fetch the value of the pointer's Offset Field 30103 from memory. It then saves Logical Descriptor 27116's offset in the output register belonging to OFFALU 20242 and places the value of the pointer's Offset Field 30103 in the offset field of Logical Descriptor 27116 which it received as an argument. The pointer-to-descriptor microcode then saves Logical Descriptor 27116 indicating the pointer's location by storing Logical Descriptor 27116's AON and offset (obtained from OFFALU 20242) in a register in the GRF 10354 frame being used by the invocation of the pointer-to-descriptor microcode. Next, the microcode adds 40 to the offset stored in OFFALU 20242, thereby obtaining the address of NR Field 30109, and uses the address to fetch and read NR Field 30109 and Format Code Field 30113. The course of further processing is determined by the values of these fields. If NR Field 30109 indicates a resolved pointer, there are four cases, as determined by the value of Format Code Field 30113:

- Format code field = 0: The pointer is a null pointer.
- Format code field = 1: The pointer is a UID pointer.
- Format code field = 2: The pointer is an intra-object pointer.
- Any other value of the format code field: The pointer is invalid.

In the first case, the microcode sets all fields of the argument to 0; in the second, it fetches the value of UID Field 30115 from memory and invokes LAR microcode (explained in the discussion of objects), which translates the UID to the AON associated with it. The AON is then loaded into the argument's AON field. In the third case, the AON of Logical Descriptor 27116 for the pointer's location and the pointer's AON are the same, so the argument already contains the translated pointer. In the fourth case, the microcode performs a call to a pointer fault-handling Procedure 602 which handles invalid pointer faults, passing saved Logical Descriptor 27116 for the pointer as an argument. Procedure 602 which handles the fault must return a resolved pointer to the microcode, which then converts it to a Logical Descriptor 27116 as described above.

c. Descriptor to Pointer Conversion

Descriptor to pointer conversion is the reverse of pointer to descriptor conversion with resolved pointers. The operation must be performed whenever a resolved pointer is moved from an FU 10120 register into MEM 10112. The operation takes two arguments: a Logical Descriptor 27116 which specifies the address to which the pointer is to be written, and a Logical Descriptor 27116 whose AON and offset fields specify the location contained in the pointer. There are two cases: intra-object pointers and UID pointers. Both kinds of pointers have values in Offset Field 30103, so the descriptor-to-pointer microcode first writes the second argument's offset to location specified by the first argument's Logical Descriptor 27116. The next step is to determine whether the pointer is an intra-object pointer or a UID pointer. To do so, the microcode compares the arguments' AONs. If they are the same, the pointer points to a location in the object which contains it, and is therefore an intra-object pointer. Since UID Field 30115 of an intra-object pointer is meaningless, the only step remaining for intra-object pointers is to set Flags and Format Field 30105 to the binary representation of 2, which sets all bits but bit 46 to 0, and thereby identifies the pointer as a resolved intra-object pointer.

With UID pointers, the descriptor-to-pointer microcode sets Flags and Format Field 30105 to 1, thereby identifying the pointer as a resolved UID pointer, and calls a KOS LAR microroutine (explained in detail in the discussion of objects) which converts the first argument's AON to a UID and places the result UID in the current frame. When the KOS AON to UID conversion microroutine returns, the descriptor-to-pointer microcode writes the UID to the converted pointer's UID Field 30115.

B. Namespace and the S-Interpreters (Figs. 303—307)

Namespace and the S-Interpreter both interpret information contained in Procedure Objects 608. Consequently, the discussion of these components of CS 10110 begins with an overview of those parts of Procedure Object 606 relevant to Namespace and the S-interpreters, and then explains Namespace and the S-interpreters in detail.

a. Procedure Object 606 Overview (Fig. 303)

Figure 303 represents those portions of Procedure Object 608. Fig. 303 expands information contained in Fig. 103; Fields which appear in both Figures have the number of Fig. 103. Portions of Procedure Object 608 which are not discussed here are dealt with later in the discussion of Calls and Returns. The most important part of a Procedure Object 608 for these systems is Procedure Environment Descriptor (PED) 30303. A Procedure 602's PED 30303 contains the information required by Namespace and the S-interpreter to locate and parse Procedure 602's code and interpret its Names. A number of Procedures 602 in a Procedure Object 608 may share a PED 30303. As will be seen in the discussion of Calls, the fact that a Procedure 602 shares a PED 30303 with the Procedure 602 that invokes it affects the manner in which the Call is executed.

The fields of PED 30303 which are important to the present discussion are three fields in Header 30304: K Field 30305, LN Field 30307, and SIP Field 30309, and three of the remaining fields: NTP Field 30311, SDPP Field 30313, and PBP Field 30315.

- K Field 30305 indicates whether the Names in the SInS of Procedures 602 which share PED 30303 have 8, 12, or 16 bits.
- LN Field 30307 contains the Name which has the largest index of any in Procedure 602's Name Table 10350.
- 5 — SIP Field 30309 is a UID pointer to the object which contains the S-interpreter for Procedure 602's S-Language.
- NTP Field 30311 is an object-relative pointer to the beginning of Procedure 602's Name Table 10350.
- SDPP Field 30313 is a pointer which is resolved to the location of static data used by Procedures 602 to which PED 30303 belongs when one of Procedures 602 is invoked by a given Process 610. The resolved pointer corresponding to SDPP 30313 is the SDP ABP.
- 10 — PBP Field 30315 contains the PBP ABP for invocations of Procedures 602 to which PED 30303 belongs. The PBP ABP is used to calculate locations inside Procedure Object 608.

Other areas of interest in Procedure Object 608 are Literals 30301 and Static Data Prototype (SDPR) 30317. Literals 30301 contains literal values, i.e., values in Procedure 602 which are known at compile time and will not change during program execution. SDPR 30317 may contain any of the following: pointers to external routines and to static data contained in other objects, information required to create static data for a Procedure 602, and in some cases, the static data itself. Pointers in SDPR 30317 may be either resolved or non-resolved.

In the present embodiment, Binder Area 30323 is also important. Binder Area 30323 contains information which allows unresolved pointers contained in Procedure Object 608 to be resolved. Unresolved pointers other than SDPP 30313 in Procedure Object 608 all contain locations in Binder Area 30323, and the specified location contains the information required to resolve the pointer.

Fig. 303 contains arrows showing the locations in Procedure Object 608 pointed to by NTP Field 30311, SDPP Field 30313, and PBP Field 30315. NTP Field 30311 points to the beginning of Name Tables 10350, and thus a Name's Name Table Entry can be located by adding the Name's value to NTP Field 30311. PBP Field 30315 points to the beginning of Literals 30301, and consequently, the locations of Literals and the locations of SInS may be expressed as offsets from the value of PBP Field 30315. SDPP Field 30313 points to the beginning of SDPR 30317. As will be explained in detail in the discussion of Calls, when a procedure 602 has static data, the SDP ABP is derived from SDPP Field 30313.

b. Namespace

The Namespace component of CS 10110 locates SInS belonging to a procedure and fetches them from memory to FU 10120, parses SInS into SOPs and Names, and performs Resolve and Evaluation operations on Names. The Resolve operation translates a Name into a Logical Descriptor 27116 for the data represented by the Name, while the Evaluation operation obtains the data itself. The Evaluation operation does so by performing a Resolve operation and then using the resulting Logical Descriptor 27116 to fetch the data. Since the Evaluation and Resolve operations are the most complicated, the discussion begins with them.

1. Name Resolution and Evaluation

Name Resolution and Evaluation translate Names into Logical Descriptors 27116 by means of information contained in the Names' NTEs, and the NTEs define locations in terms of Architectural Base Registers. Consequently, the following discussion will first describe Name Table Entries and Architectural Base Pointers and then the means by which Namespace translates the information contained in the Name Table Entries and Architectural Base Pointers into Logical Descriptors 27116.

2. The Name Table (Fig. 304)

As previously mentioned, Name Tables 10350 are contained in Procedure Objects 608. Name Tables 10350 contain the information required to translate Names into Logical Descriptors 27116 for the operands represented by the Names. Each Name has as its value the number of a Name Table Entry. A Name's Name Table Entry is located by multiplying the Name's value by the size of a short Name Table Entry and adding the product to the value in NTP Field 30311 of PED 30303 belonging to Procedure 602 which contains the SIn.

The Name Table Entry contains length and type information for the data item specified by the Name, and represents the data item's location as a displacement from a known location, termed the base. The base may be a location specified by an ABP, a location specified by another Name, or a location specified by a pointer. In the latter case, the pointer's location may be specified in terms of an ABP or as a Name.

Fig. 304 is a detailed representation of a Name Table Entry (NTE) 30401. There are two kinds of NTEs 30401: Short NTEs 30403 and Long NTEs 30405. Short NTEs 30403 contain 64 bits; Long NTEs 30405 contain 128 bits. Names that represent scalar data items whose displacements may be expressed in 16 bits have Short NTEs 30403; Names that represent scalar data items whose displacements require more than 16 bits and Names that represent array elements have Long NTEs 30405.

A Short NTE 30403 has four main fields, each 16 bits in length:

- Flags and Format Field 30407 contains flags and format information which specify how Namespace is to interpret NTE 30401.

EP 0 067 556 B1

- Base Field 30425 indicates the base to which the displacement is to be added to obtain the location of the data represented by the Name. Base Field 30425 may represent the location in four ways: by means of an ABP by means of a Name, by means of a pointer located by means of an ABP, and by means of a pointer located by means of a Name.
 - Length Field 30435 represents the length of the data. The length may be a literal value or a Name. If it is a Name, the Name resolves to a location which contains the data item's length.
 - Displacement Field 30437 contains the displacement of the beginning of the data from the base specified in Field 30425. The displacement is a signed integer value.
- Long NTEs 30405 have four additional fields, each 16 bits long: Two of the fields, Index Name Field 30441 and IES Field 30445 are used only in NTEs 30401 for Names that represent arrays.
- Displacement Extension Field 30439 is used in all Long NTEs 30405. If the displacement value in Field 30437 has less than 16 bits, Displacement Extension Field 30439 contains sign bits, i.e., the bits in the field are set to 0 when the displacement is positive and 1 when the displacement is negative. When the displacement value has more than 16 bits, Displacement Extension Field 30439 contains the most significant bits of the displacement value as well as sign bits.
 - Index Name Field 30441 contains a Name that represents a value used to index an element of an array.
 - Field 30443 is reserved.
 - IES Field 30445 contains a Name or Literal that specifies the size of an element in an array. The value represented by this field is used together with the value represented by Index Name Field 30441 to locate an element of an array.
- As may be seen from the above, the following fields may contain names: Base Field 30425, Length Field 30435, Index Name Field 30441, and IES Field 30445.
- Two fields in NTE 30401 require further consideration: Flags and Format Field 30407 and Base Field 30425. Flags and Format Field 30407 has three subfields: Flags Field 30408, FM Field 30421, and Type Field 30423. Turning first to Flags Field 30408, the six flags in the field indicate how Namespace is to interpret NTE 30401. The flags have the following meanings when they are set:
- Long NTE Flag 30409: NTE 30401 is a Long NTE 30405.
 - Length is a Name Flag 30411: Length Field 30435 contains a Name.
 - Base is a Name Flag 30413: Base Field 30425 contains a Name instead of the number of an ABP.
 - Base Indirect Flag 30415: Base Field 30425 represents a pointer, and the location represented by NTE 30401 is to be calculated by obtaining the pointer's value and adding the value contained in Displacement Field 30437 and Displacement Extension Field 30439 to the pointer's offset.
 - Array Flag 30417: NTE 30401 represents an array.
 - IES is a Name Flag 30419: IES Field 30445 contains a Name that represents the IES value.
- Several of these flags may be set in a given NTE 30401. For example, an entry for an array element that was referenced via a pointer to the array which in turn was represented by a Name, and whose IES value was represented by a Name, would have Flags 30409, 30413, 30415, 30417, and 30419 set.
- FM Field 30421 indicates how the data represented by the Name is to be formatted when it is fetched from memory. The value of FM Field 30421 is placed in FIU Field 27107 of Logical Descriptor 27116 produced from NTE 30401. The two bits allow for four possibilities:

Setting	Meaning
00	right justify, zero fill
01	right justify, sign fill
10	left justify, zero fill
11	left justify, ASCII space fill

The four bits in Type Field 30423 are used by compilers for language-specific type information. The value of Type Field 30423 is placed in Type Field 27109 of Logical Descriptor 27116 produced from NTE 30401.

Base Field 30425 may have either Base is an ABP Format 30427 or Base is a Name Format 30432. The manner in which Base Field 30425 is interpreted depends on the setting of Base is a Name Flag 30413 and Base Indirect Flag 30415. There are four possibilities:

EP 0 067 556 B1

Field Settings

	Base is a Name	Base Indirect	Meaning
5	0	0	ABP Format locates base directly.
	0	1	ABP Format locates a pointer which is the base.
10	1	0	Base is Name Format locates base when Name is resolved.
15	1	1	Base is Name Format locates a pointer when Name is resolve and the pointer is the base.

As indicated by the above table, Base Field 30425 is interpreted as having Base is ABP Format 30427 when Base is a Name Flag 30411 is not set. In Base is ABP Format 30427, Base Field 30425 has two subfields: ABP Field 30429 and Pointer Locator Field 30431. The latter field has meaning only when Base Indirect Flag 30415 is set. ABP Field 30429 is a two-bit code which indicates the ABP. The settings and their meanings are the following:

Setting	APB
00	FP
01	Unused
10	SDP
11	PBP

The ABPs are discussed below. When Base Indirect Flag 30415 is set to 1 and Base is a Name Flag 30413 is set to 0, the remaining 14 bits of the Base Field in ABP Format are interpreted as Pointer Locator Field 30413. When so interpreted, Pointer Locator Field 30413 contains a signed integer, which, when multiplied by 128, gives the displacement of a pointer from the ABP specified in ABP Field 30429. The value of this pointer is then the base to which the displacement is added.

Base Field 30425 is interpreted as having Base is a Name Format 30432 when Base is a Name Flag 30413 is set to 1. In Base is a Name Format 30432, Base Field 30425 contains a Name. If Base Indirect Flag 30415 is not set, the Name is resolved to obtain the Base. If Base Indirect Flag 30415 is set, the name is evaluated to obtain a pointer value, and that pointer value is the Base.

3. Architectural Base Pointers (Figs. 305, 306)

If Base is a Name Flag 30413 belonging to a NTE 30401 is not set, Base Field 30425 specifies one of the three ABPs in CS 10110:

- PBP specifies a location in Procedure Object 608 to which displacements may be added to obtain the locations of Literals and SInS.
- SDP specifies a location in a Static Data Block for an invocation of a Procedure 602 to which displacements may be added to obtain the locations of static data and linkage pointers to Procedures 602 contained in other Procedure Objects 608 and static data.
- FP specifies a location in the MAS frame belonging to Procedure 602's current invocation to which displacements may be added to obtain the location of local data and linkage pointers to arguments.

Each time a Process 610 invokes a Procedure 602, Call microcode saves the current values of the ABPs on Secure Stack 10336, calculates the values of the ABPs for the new invocation, and places the resulting Logical Descriptors 27116 in FU 10120 registers, where they are accessible to Namespace microcode.

Call microcode calculates the ABPs as follows: PBP is obtained directly from PBP Field 30315 in PED 30303 belonging to the Procedure 602 being executed. All that is required to make it into a Logical Descriptor 27116 is the addition of the AON for Procedure Object 608's UID.

SDP is obtained by performing a pointer-to-descriptor translation on SDPP Field 30313. FP, finally, is provided by the portion of Call microcode which creates the new MAS 502 frame for the invocation. As is described in detail in the discussion of Call, the Call microcode copies linkage pointers to the invocation's actual arguments onto MAS 502, sets FP to point to the location following the last actual argument, and then allocates storage for the invocation's local data. Positive displacements from FP thus specify locations

EP 0 067 556 B1

in the local data, while negative offsets specify linkage pointers.

a.a. Resolving and Evaluating Names (Fig. 305)

The primary operations performed by Namespace are resolving names and evaluating them. A Name has been resolved when Namespace has used the ABPs and information contained in the Name's NTE 30401 to produce a Logical Descriptor 27116 for the Name; a name has been evaluated when Namespace has resolved the Name, presented the resulting Logical Descriptor 27116 for the Name to memory, and obtained the value of the data represented by the Name from memory.

The resolve operation has three parts, which may be performed in any order:

- Obtaining the Base from Base Field 30425 of the Name's NTE 30401.
- Obtaining the displacement.
- Obtaining the length from Length Field 30435.

Obtaining the length is the simplest of the operations: if Length in a Name Flag 30411 is set, the length is the value obtained by evaluating the Name contained in Length Field 30435; otherwise, Length Field 30435 contains a literal value and the length is that literal's value.

There are four ways in which the Base may be calculated. Which is used depends on the settings of Base is a Name Flag 30413 and Base Indirect Flag 30415:

- Both Flags 0: the ABP specified in ABP Field 30429 is the Base.
- Base is a Name Flag 30413 0 and Base Indirect Flag 30415 1: The Base is the location contained in the pointer specified by ABP Field 30429 and pointer Locator Field 30431.
- Base is a Name Flag 30413 1 and Base Indirect Flag 30415 0: The Base is the location obtained by resolving the Name in Base Field 30425.
- Both Flags 1: The Base is the location obtained by evaluating the Name in Base Field 30425.

The manner in which Namespace calculates the displacement depends on whether NTE 30401 represents a scalar data item or an array data item. In the first case, Namespace adds the value contained in Displacement Field 30437 and Displacement Extension Field 30439 to the location obtained for the Base; in the second case, Namespace evaluates Index Name Field 30441 and IES Field 30445, multiplies the resulting values together, and adds the product to the value in Displacement Field 30437 in order to obtain the displacement.

If any field of a NTE 30401 contains a Name, Namespace obtains the value or location represented by the Name by performing a Resolve or Evaluation operation on it as required. As mentioned in the discussion of NTEs 30401, flags in Flags Field 30408 indicate which fields of an NTE 30401 contain Names. Since the NTE 30401 for a Name used in another NTE 30401 may itself contain Names, Namespace performs the Resolve and Evaluation operations recursively.

b.b. Implementation of Name Evaluation and Name Resolve in CS 10110

In the present embodiment, the Name Evaluation and Resolve operations are carried out by FU 10120 microcode Eval and Resolve commands. Both commands require two pieces of information: a register in the current frame of SR portion 10362 of GRF 10354 for receiving Logical Descriptor 27116 produced by the operation, and the source of the Name which is to be resolved or evaluated. Both Resolve and Eval may choose between three sources: Parser 20264, Name Trap 20254, and the low-order 16 bits of the output register for OFFALU 20242. Resolve may specify current frame registers 0, 1, or 2 for Logical Descriptor 27116, and Eval may specify current frame registers 0 or 1. At the end of the Resolve operation, Logical Descriptor 27116 for the data represented by the Name is in the specified SR 10362 register and at the end of the Evaluation operation, Logical Descriptor 27116 is in the specified SR 10362 register and the data's value has been transferred via MOD Bus 10114 to EU 10122's OPB 20322.

The execution of both Resolve and Eval commands always begin with the presentation of the Name to Name Cache 10226. The Name presented to Name Cache 10226 is latched into Name Trap 20254, where it is available for subsequent use by Name Resolve microcode.

If there is an entry for the Name in Name Cache 10226, a name cache hit occurs. For Names with NTEs 30401 fulfilling three conditions, the Name Cache 10226 entry for the Name is a Logical Descriptor 27116 for the data item represented by the Name. The conditions are the following:

- NTE 30401 contains no Names.
- Length Field of NTE 30401 specifies a length of less than 256 bits.
- If Base is Indirect Flag 30415 is set, Pointer Displacement Field 30431 must have a negative value, indicating that the base is a linkage pointer.

Logical Descriptor 27116 can be encached in this case because neither the location nor the length of the data represented by the Name can change during the life of an invocation of Procedure 602 to which the Name belongs. If the Name Cache 10226 entry for the Name is a Logical Descriptor 27116, the hit causes Name Cache 10226 to place Logical Descriptor 27116 in the specified SR 10362 register. In all other cases, the Name Cache 10226 entry for the Name does not contain a Logical Descriptor 27116, and a hit causes Name Cache 10226 to emit a JAM signal. The JAM signal invokes microcode which uses information stored in Name Cache 10226 to construct Logical Descriptor 27116 for the data item represented by the Name. JAMS are explained in detail below.

If there is no entry for the Name in Name Cache 10226, a Name Cache Miss occurs, and Name Cache 10226 emits a cache miss JAM signal. The Name Resolve microroutine invoked by the cache miss JAM

signal constructs an entry in Name Cache 10226 from the Name's NTE 30401, using FU 10120's DESP 20210 to perform the necessary calculations. When it is finished, the cache miss microcode leaves a Logical Descriptor 27116 for the Name in the specified SR 10362 register and returns.

5 The Resolve operation is over when Logical Descriptor 27116 has been placed in the specified GRF 10354 register; the Evaluation operation continues by presenting Logical Descriptor 27116 to Memory Reference Unit 27017, which reads the data represented by Logical Descriptor 27116 from memory and places it on OPB 20322. The memory reference may result in Protection Cache 10234 misses and ATU 10228 misses, as well as protection faults and page faults, but these are handled by means of event signals and are therefore invisible to the Evaluation operation.

10 Name Cache 10226 produces 15 different JAM signals. The signal produced by a JAM depends on the following: whether the operation is a Resolve or an Eval, which register Logical Descriptor 27116 is to be placed in, whether a miss occurred, and in the case of a hit, which register in the Name Cache 10226 entry for the Name was loaded last. From the point of view of the behavior of the microcode invoked by the JAM, the last two factors are the most important. Their relation to the microcode is explained in detail below.

15 In the present embodiment, all entries in Name Cache 10226 are invalidated when a Procedure 602 calls another Procedure 602. The invalidation is required because Calls always change the value of FP and may also change the values of SDP and PBP, thereby changing the meaning of NTEs 30401 using displacements from ABPs. Entries for Names in invoked Procedure 602 are created and loaded into Name Cache 10226 when the Names are evaluated or resolved and a cache miss occurs.

20 The following discussion will first present Name Cache 10226 as it appears to the microprogrammer and then explain in detail how Name Cache 10226 is used to evaluate and resolve Names, how it is loaded, and how it is flushed.

c.c. Name Cache 10226 Entries (Fig. 306)

25 The structure and the physical behavior of Name Cache 10226 was presented in the discussion of FU 10120 hardware; here, the logical structure of Name Cache 10226 entries as they appear to the microprogrammer is presented. To the microprogrammer, Name Cache 10226 appears as a device which, when presented a Name on NAME Bus 20224, always provides the microprogrammer with a Name Cache 10226 entry for the Name consisting of four registers. The microprogrammer may read from or write to any one of the four registers. When the microprogrammer writes to the four registers, the action taken by Name Cache 10226 when a hit occurs on the Name associated with the four registers depends on which of the registers has most recently been loaded. The means by which Name Cache 10226 associates a Name with the four registers, and the means by which Name Cache 10226 provides registers when it is full are invisible to the microprogrammer.

35 Fig. 306 illustrates Name Cache Entry 30601 for a Name. The four Registers 30602 in Name Cache Entry 30601 are numbered 0 through 3, and each Register 30602 has an AON, offset, and length field like those in GRF 10354 registers, except that some flag bits in GRF 10354 register AON fields are not included in Register 30602 fields, and the length field in Register 30602 is 8 bits long. As is the case with GRF 10354 registers, the microprogrammer can read or write individual fields of Register 30602 or entire Register 30602. Name Cache Entry 30601 is connected via DB 27021 to DESP 20210, and consequently, the contents of a GRF 10354 register may be obtained from or transferred to a Register 30602 or viceversa. When the contents of a Register 30602 have been transferred to a GRF 10354 register, the contents may be processed using OFFALU 20242 and other arithmetic-logical devices in DESP 20210.

45 d.d. Name Cache 10226 Hits

When a Name is presented to Name Cache 10226 and Name Cache 10226 has a Name Cache Entry 30601 containing information about the Name, a name cache hit occurs. On a hit, Name Cache 10226 hardware always loads the contents of Register 30602 0 of the Name's Name Cache Entry 30601 into the GRF 10354 register specified in the Resolve or Eval microcommand. In addition, a hit may result in the invocation of microcode via a JAM:

- The JAM may invoke special microcode for resolving Names of array elements whose NTEs 30401 allow certain hardware accelerations of index calculations.
- The JAM may invoke general name resolution microcode which produces a Logical Descriptor 27116 from the contents of Name Cache Entry 30601.

55 Whether the hit produces a JAM, and the kind of JAM it produces, are determined by the last Register 30602 to be loaded when Name Cache Entry 30601 was created by Name Cache Miss microcode. If Register 30602 0 was the last to be loaded, no JAM occurs; if Register 30602 1 was loaded last, the JAM for special array Name resolution occurs; if Register 30602 2 or 3 was loaded last, the JAM for general Name resolution occurs.

60 As may be inferred from the above, Name Cache 10226 hardware defines the manner in which Name Cache Entries 30601 are loaded for the first two cases. In the first case, Name Cache Register 30602 0 must contain Logical Descriptor 27116 for the Name's data. As already mentioned, the Name's NTE 30401 must therefore describe data whose location and length does not change during an invocation and whose length is less than 256 bits. Name Cache 10226 hardware also determines the form of Name Cache Entries 30601 for encachable arrays. An encachable array NTE 30401 is an array NTE 30401 which fills the following

EP 0 067 556 B1

conditions:

- The only Name contained in array NTE 30401 is in Index Name Field 30441.
- NTE 30401 for the index Name fills the conditions for scaler NTEs 30401 for which Logical Descriptors 27116 may be encached.
- 5 — The value in IES Field 30445 is no greater than 128 and a power of 2.
- Array NTE 30401 otherwise fills the conditions for scaler NTEs 30401 for which Logical Descriptors 27116 may be encached.

In the present embodiment, the encachable array entry uses registers 0, 1, and 2 of Name Cache Entry 30601 for the name:

10

Register	Contents		
	AON	OFFSET	LENGTH
0	Logical Descriptor 27116 for the index Name		
1	0	IES power of 2	unused
20	2	Logical Descriptor 27116 for the array	

When a hit for this type of entry occurs, the resulting JAM signal does two things: it invokes encachable array resolve microcode and it causes the index Name's Logical Descriptor 27116 to be presented to Memory Reference Unit 27017 for a read operation which returns the value of the data represented by the index Name to an accumulator in OFFALU 20242. The encachable array resolve microcode then uses the Name that caused the JAM, latched into Name Trap 20254, to locate Register 30602 2 of Name Cache Entry 30601 for the Name, writes the contents of Register 30602 2 into the GRF register specified by the Resolve or Eval microcommand, obtains the product of the IES value and the index value by shifting the index value left the number of times specified by the IES exponent in Register 30602 1, adds the result to the offset field of the GRF 10354 register containing the array's Logical Descriptor 27116, thus obtaining Logical Descriptor 27116 for the desired array element, and returns.

For the other cases, the manner in which Name Cache Entries 30601 are loaded and processed to obtain Logical Descriptors 27116 is determined by the microprogrammer. The JAM signal which results if a Name Cache Entry 30601 is neither a Logical Descriptor 27116 nor an encachable array entry merely invokes a microcode routine. The microcode routine uses the Name latched into Name Trap 20254 to locate the Name's Name Cache Entry 30601 and then reads tag values in Name Cache Entry 30601 to determine how the information in Name Cache Entry 30601 is to be translated into a Logical Descriptor 27116. The contents of Name Cache Entries 30601 for the other cases have two general forms: one for NTEs 30401 with Base is Indirect Flag 30415 set, and one for NTEs in which it is not set. The first general form looks like this:

Register	Contents		
	AON	OFFSET	LENGTH
0	ABP AON	tag/length	unused
1	0	index name/IES	unused
2	0	unused	unused
3	0	data displacement from loc. specified by pointer	unused

Register 30602 0 contains the AON of the ABP. Register 30602 0's offset field contains two items: the tag, which contains Flags Field 30408 of NTE 30401 along with other information, and which determines how Name Resolve microcode interprets the contents of Name Cache Entry 30601, and a value or Name for the length of the data item. Register 30602 1 is used only if the Name represents a data item in an array. It then contains the Name from Index Field 30441 and the Name or value from IES Field 30445. The offset field of Register 30602 3 contains the sum of the offset indicated by NTE 30401's ABP and of the displacement indicated by NTE 30401.

The second format, used for NTEs 30401 whose bases are obtained from pointers or by resolving a Name, looks like this:

65

EP 0 067 556 B1

	Registers	Contents		
		AON	OFFSET	LENGTH
5	0	0	tag/length	unused
	1	0	index name/IES	unused
10	2	0	FM and type bits/ base field	unused
15	3	0	data displacement from loc. specified by pointer or name	unused

In this form, the location of the Base must be obtained either by evaluating a pointer or resolving a Name. Hence, there is no field specifying the Base's AON. Otherwise, Registers 30602 0 and 1 have the same contents as in the previous format. In Register 30602 2, the offset field contains Name Table Entry 30401's FM Field 30421 and Type Field 30423 and Base Field 30425. The Offset Field of Register 30602 2 contains the value of Name Table Entry 30401 Displacement Fields 30437 and 30439.

As in Name Table Entries 30401, the index must be represented by a Name, and length, IES, and Base may be represented by Names. If a field of Name Cache Entry 30601 contains a Name, a flag in the tag indicates that fact, and Name Resolve microcode performs an Eval or Resolve operation on it as required to obtain the value or location represented by the name.

The microcode which resolves Name Cache Entries 30601 of the types just described uses the general algorithms described in the discussion of Name Table Entries 30401, and is therefore not discussed further here.

e.e. Name Cache 10226 Misses

When a Name is presented to Name Cache 10226 and there is no Name Cache Entry 30601 for the Name, a name cache miss occurs. On a miss Name Cache 10226 hardware emits a JAM signal which invokes name cache miss microcode. The microcode obtains the Name which caused the miss from Name Trap 20254 and locates the Name's NTE 30401 by adding the Name to the value of NTP 30311 from PED 30303 for Procedure 602 being executed. As will be explained in detail later, when a Procedure 602 is called, the Call microcode places the AON and offset specifying the NTP's location in a register in GR's 10360. Using the information contained in the Name's NTE 30401, the Cache Miss microcode resolves the Name and constructs a Name Cache Entry 30601 for it. As described above, the microcode determines the method by which it resolves the Name and the form of the Name's Name Cache Entry 30601 by reading Flags Field 30408 in the Name's NTE 30401. Since the descriptions of the Resolve operation, the micromachine, Name Cache 10226, and the formats of Name Cache Entries 30601 are sufficient to allow those skilled in the art to understand the operations performed by Cache Miss microcode, no further description of the microcode is provided.

f.f. Flushing Name Cache 10226

As described in the discussion of Name Cache 10226 hardware, hardware means, namely VALS 24068, exist which allow Name Cache Entries 30601 to be invalidated. Name Cache Entries 30601 may be invalidated singly, or all entries in Name Cache 10226 may be invalidated by means of a single microcommand. The latter operation is termed name cache flushing. In the present embodiment, Name Cache 10226 must be flushed when Process 610 whose Virtual Processor 612 is bound to JP 10114 executes a Call or a Return and whenever Virtual Processor 612 NO is unbound from JP 10114. Flushing is required on Call and Return because Calls and Returns change the values of the ABPs and other pointers needed to resolve Names. At a minimum, a Call produces a new MAS Frame 10412, and a Return returns to a previous Frame 10412, thereby changing the value of FP. If the called Procedure 602 has a different PED 30303 from that of the calling Procedure 602, the Call or Return may also change PBP, SDP, and NTP. Flushing is required when a Virtual Processor 612 is unbound from JP 10114 because Virtual Processor 612 which is next bound to JP 10114 is bound to a different Process 610, and therefore cannot use any information belonging to Process 610 bound to the Previous Virtual Processor 612.

g.g. Fetching the I-Stream

As explained in the discussion of FU 10120 hardware, SInS are fetched from memory by Prefetcher 20260. PREF 20260 contains a Logical Descriptor 27116 for a location in Code 10344 belonging to Procedure 602 which is currently being executed. On any MO cycle, PREF 20260 can place Logical Descriptor 27116 on DB 27021, cause Memory Reference Unit 27017 to fetch 32 bits at the location specified by Logical

Descriptor 27116, and write them into INSTB 20262. When INSTB 20262 is full, PREF 20260 stops fetching SINs until Namespace parsing operations, described below, have processed part of the contents of INSTB 20262, thereby creating space for more SINs.

The fetching operation is automatic, and requires intervention from Namespace only when a SIN causes a branch, i.e., causes the next SIN to be executed to be some other SIN than the one immediately following the current SIN. On a branch, Namespace must load PREF 20260 with the location of the next SIN to be executed and cause PREF 20260 to begin fetching SINs at that location. The operation which does this is specified by the load-prefetch-for-branch microcommand. The microcommand specifies a source for a Logical Descriptor 27116 and transfers that Logical Descriptor 27116 via DB 27021 to PREF 20260. After PREF 20260 has thus been loaded, it begins fetching SINs at the specified location. Since any SINs still in INSTB 20262 have been rendered meaningless by the branch operation, the first SINs loaded into INSTB 20262 are simply written over INSTB 20262's prior contents. Fig. 274 contains an example of the use of the load-prefetch-for-branch microcommand.

15 h.h. Parsing the I-Stream

The I-stream as fetched from MEM 10112 and stored in INSTB 20262 is a sequence of SOPs and Names. As already mentioned, the I-stream has a fixed format: in the present embodiment, SOPs are always 8 bits long, and Names may be 8, 12, or 16 bits long. The length of Names used in a given procedure is fixed, and is indicated by the value in K Field 30305 in the Procedure 602's PED 30303. The Namespace parsing operations obtain the SOPs and Names from the I-stream and place them on NAME Bus 20224. The SOPs are transferred via this bus to the devices in SOP Decoder 27003, while the Names are transferred to Name Trap 20254 and Name Cache 10226 for Resolve and Evaluation operations as described above. As the parsing operations obtain SOPs and Names, they also update the three program counters CPC 20270, EPC 20274, and IPC 20272. The values in these three counters are offsets from PBP which point to locations in Code 10344 belonging to Procedure 602 being executed. CPC 20270 points to the I-stream syllable currently being parsed, so it is updated on every parsing operation. EPC 20274 points to the beginning of the last SIN executed by JP 10114, and IPC 20272 points to the beginning of the current SIN, so these program counters are changed only at the beginning of the execution of an SIN, i.e., when a SOP is parsed.

As described in the discussion of FU 10120 hardware, in the current implementation, parsing consists physically of reading 8 or 16 bits of data from a location in INSTB 20262 identified by a pointer for INSTB 20262 which is accessible only to the hardware. As data is read, the hardware increments the pointer by the number of bits read, wrapping around and returning to the beginning of INSTB 20262 if it reaches the end. At the same time that the hardware increments the pointer, it increments CPC 20270 by the same number of bits. As previously mentioned, CPC 20270 contains the offset from PBP of the SOP or Name being currently parsed, thus coordinating the reading of INSTB 20262 with the reading of Procedure 602's Code 10344.

The number of bits read depends on whether Parser 20264 is reading an SOP or a Name, and in the latter case, by the syllable size specified for the Name. The syllable size is contained in CSSR 24112. On a Call to a Procedure 602 which has a different PED 30303 from that of the calling procedure, the Call microcode loads the value contained in K Field 30305 into CSSR 24112.

Namespace's parsing operations are performed by separate microcommands for parsing SOPs and Names. There is a single microcommand for parsing S-operations: parse-op-stage. The microcommand obtains the next eight bits from INSTB 20262, places the bits onto NAME Bus 20224, and latches them into LOPCODE Register 24212. It also updates EPC 20274 and IPC 20272 as required at the beginning of an SIN: EPC 20274 is set to IPC 20272's former value, and IPC 20272 is set to CPC 20270's value. At the end of the operation, CPC 20270 is incremented by 8. Since the parsing of an SOP always occurs as the first operation in the interpretation of an SIN, the parse-op-stage command is generally combined with a dispatch fetch command. As will be explained below, the latter command interprets the S-operation as an address in FDISP 24218, and FDISP 24218 in turn produces an address in FUSITT 11012. The latter address is the location of the beginning of the SIN microcode for the SIN.

50 There are two microcommands for parsing Names:

parse_k_load_epc and parse_k_dispatch_ebox. Both commands obtain a number of bits from INSTB 20262 and place them on NAME Bus 20214. With both microcommands, the syllable size, K, stored in CSSR 24112, determines the number of bits obtained from INSTB 20262. Both commands also increment CPC by the value stored in CSSR 24112. In addition, parse_k_load_epc sets EPC to IPC's value, while parse_k_dispatch_ebox also dispatches EU 10122, i.e., interprets the SOP saved in LOPCODE 24210 as an address in EDISP 24222, which in turn contains an address in EU EUSITT 20344. The EU EUSITT 20344 address is passed via EUDIS Bus 20206 to COMQ 20342 in EU 10122.

60 c. The S-Interpreters (Fig. 307)

CS 10110 does not assign fixed meanings to SOPs. While all SOPs are 8 bits long, a given 8 bit SOP may have one meaning in one S-Language and a completely different meaning in another S-Language. The semantics of an S-Language's S-operations are determined completely by the S-interpreter for the S-Language. Thus, in order to correctly interpret an S-operation, CS 10110 must know what S-interpreter it is to use. The S-interpreter is identified by a UID pointer with offset 0 in SIP Field 30309 of PED 30303 for

EP 0 067 556 B1

Procedure 602 that CS 10110 is currently executing. In the present embodiment, the UID is the UID of a microcode object which contains FU 10120 microcode. When loaded into FUSITT 11012, the microcode interprets SOPs as defined by the S-Language to which the SOP belongs. In other embodiments, the UID may be the UID of a Procedure Object 608 containing Procedures 602 which interpret the S-Language's SOPs, and in still others, the S-interpreter may be contained in a PROM and the S-interpreter UID may not specify an object, but may serve solely to identify the S-interpreter.

When a Procedure 602 executes an SIN on JP 10114, CS 10110 must translate the value of SIP Pointer 30309 for Procedure 602 and the S-instruction's SOP into a location in the microcode or high-level language code which makes up the S-interpreter. The location obtained by the translation is the beginning of the microcode or high-level language code which implements the SIN. The translation of an SOP together with SIP Pointer 30309 into a location in the S-interpreter is termed dispatching. Dispatching in the present embodiment involves two primary components: a table in memory which translates the value of SIP Pointer 30309 into a small integer called the Dialect Number, and S-operation Decoder Portion 27003 of the FU 10120 micromachine. The following discussion will first present the table and explain how an SIP Pointer 30309 is translated into a Dialect Number, and then explain how the Dialect Number and the SOP together are translated into locations in FUSITT 11012 and EUSITT 20344.

1. Translating SIP Into a Dialect Number (Fig. 307)

In the present embodiment, all S-interpreters in CS 10110 are loaded into FUSITT 11012 when CS 10110 begins operation and each S-interpreter is always placed in the same location. Which S-interpreter is used to interpret an S-Language is determined by a value stored in dialect register RDIAL 24212. Consequently, in the present embodiment, a Call to a Procedure 602 whose S-interpreter differs from that of the calling Procedure 602 must translate the UID pointer contained in SIP Field 30309 into a Dialect Number.

Fig. 307 represents the table and microcode which performs this translation in the present embodiment. S-interpreter Translation Table (STT) 30701 is a table which is indexed by small AONs. Each STT Entry (STTE) 30703 has two fields: an AON Field 30705 and a Dialect Number Field 30709. Dialect Number Field 30709 contains the Dialect Number for the S-interpreter object whose AON is in AON Field 30705.

When CS 10110 begins operation, each S-interpreter object is wired active and assigned an AON small enough to serve as an index in STT 30701. By convention, a given S-interpreter object is always assigned the same AON and the same Dialect Number. The AON is placed in AON Field 30705 of STTE 30703 indexed by the AON, and the Dialect Number is placed in Dialect Number Field 30709. Since the S-interpreter objects are wired active, these AONs will never be reassigned to other objects.

On a Call which requires a new S-interpreter, Call microcode obtains the new SIP from SIP Field 30309, calls KOS LAR microcode to translate its UID to its AON, uses the AON to locate the S-interpreter's STTE 30703, and places the value of Dialect Number Field 30709 into RDIAL 21242.

Other embodiments may allow S-interpreters to be loaded into FUSITT 11012 at times other than system initialization, and allow S-interpreters to occupy different locations in FUSITT 11012 at different times. In these embodiments, STT 30701 may be implemented in a manner similar to the implementations of AST 10914 or MHT 10716 in the present embodiment.

2. Dispatching

Dispatching is accomplished by Dispatch Files 27004. These files translate the values provided by RDIAL 24212 and the SOP of the S-instruction being executed into the location of microcode for the SIN specified by the S-operation in the S-interpreter specified by the value of RDIAL 24212. The present embodiment has three dispatch files: FDISP 24218, FALG 24220, and EDISP 24222. FDISP 24218 and FALG 24220 translate S-operations into locations of microcode which executes on FU 10120; EDISP 24222 translates S-operations into locations of microcode which executes on EU 10122. The difference between FDISP 24218 and FALG 24220 is one of speed: FDISP 24218 can translate an SOP in the same microinstruction which performs a parse_op_stage command to load the SOP into LOPCODE 24210. FALG 24220 must perform the translation on a cycle following the one in which the SOP is loaded into LOPCODE 24210. Typically, the location of the first portion of the microcode to execute an S-operation is contained in an FDISP 24218 register, the location of portions executed later is contained in an FALG 24220 register, and the location of microcode for the S-operation which executes on EU 10122 is contained in EDISP 24222.

In the present embodiment, the registers accomplish the translation from S-operation to microcode location as follows: As mentioned in the discussion of FU 10120 hardware, each Dispatch File contains 1024 registers. Each register may contain an address in an S-interpreter. As will be seen in detail later, the address may be an address in an S-interpreter's object, or it may be the address in FUSITT 11012 or EUSITT 20344 of a copy of microcode stored at an S-interpreter address. The registers in the Dispatch Files may be divided into sets of 128 or 256 registers. Each set of registers translates the SOPs for a single S-Language into locations in microcode. Which set of registers is used to interpret a given S-operation is decided by the value of RDIAL 24212; which register in the set is used is determined by the value of the S-operation. The value contained in the specified register is then the location of microcode which executes the S-instruction specified by the S-operation in the S-Language specified by RDIAL 24212.

Logically, the register addressed by the concatenated value in turn contains a 15 bit address which is

the location in the S-interpreter of the first microinstruction of microcode used to execute the S-instruction specified by the S-operation in the S-Language specified by the contents of RDIAL 24212. In the present embodiment, the microcode referred to by the address may have been loaded into FUSITT 11012 and EUSITT 20344 or it may be available only in memory. Addresses of microcode located in FUSITT 11012 and EUSITT 20344 are only eight bits long. Consequently, if a Dispatch File 27004 contains an address which requires more bits than that, the microcode specified by the address is in memory. As described in the discussion of FU 10112 hardware, addresses larger than 8 bits produce an Event Signal, and microcode invoked by the event signal fetches the microinstruction at the specified address in the S-interpreter from memory and loads it into location 0 of FUSITT 11012. The event microcode then returns, and the microinstruction at location 0 is executed. If the next microinstruction also has an address larger than 8 bits, the event signal occurs again and the process described above is repeated.

As previously mentioned, FDISP 24218 is faster than FALG 24220. The reason for the difference in speed is that FDISP registers contain only 6 bits for addressing the S-interpreter. The present embodiment assumes that all microcode addressed via FDISP 24218 is contained in FUSITT 11012. It concatenates 2 zero bits with the six bits in the FDISP 24218 register to produce an 8 bit address for FUSITT 11012. FDISP 24218 registers can thus contain the location of every fourth FUSITT 11012 register between FUSITT register 256 and FUSITT register 448. The microcode loaded into these locations in FUSITT 11012 is microcode for operations which are performed at the start of the SIN by many different SINs. For example, all SINs which perform operations on 2 operands and assign the result to a location specified by a third operand must parse and evaluate the first two operands and parse and resolve the third operand. Only after these operations are done are SINs-specific operations performed. In the present embodiment, the microcode which parses, resolves, and evaluates the operands is contained in a part of FUSITT 11012 which is addressable by FDISP 24218.

As previously mentioned, in the present embodiment, FUSITT 11012 and EUSITT 20344 may be loaded only when CS 10110 is initialized. The microcode loaded into FUSITT 11012 and EUSITT 20344 is produced by the microbinder from the microcode for the various SINs. To achieve efficient use of FUSITT 11012 and EUSITT 20344, microcode for operations shared by various S-interpreters appears only once in FUSITT 11012 and EUSITT 20344. While the SINs in different S-Languages which share the microcode have different registers in FDISP 24218, FALG 24220, or EDISP 24222 as the case may be, the registers for each of the S-instructions contain the same location in FUSITT 11012 or EUSITT 20344.

4. The Kernel Operating System

A. Introduction

Many of the unique properties of CS 10110 are produced by the manipulation of tables in MEM 10112 and Secondary Storage 10124 by programs executing on JP 10114. These programs and tables together make up the Kernel Operating System (KOS). Having described CS 10110's components and the means by which they cooperate to execute computer programs, this specification now presents a detailed account of KOS and of the properties of CS 10110 which it produces. The discussion begins with a general introduction to operating systems, then presents an overview of CS 10110's operating systems, an overview of the KOS, and detailed discussions of the implementation of objects, access control, and Processes 610.

a. Operating Systems (Fig. 401)

In CS 10110, as in other computer systems, the operating system has two functions:

- It controls the use of CS 10110 resources such as JP 10114, MEM 10112, and devices in IOS 10116 by programs being executed on CS 10110.
- It defines how CS 10110 resources appear to users of CS 10110.

The second function is a consequence of the first: By controlling the manner in which executing programs use system resources, the operating system in fact determines how the system appears to its users. Figure 401 is a schematic representation of the relationship between User 40101, Operating System 40102, and System Resources 40103. When User 40101 wishes to use a System Resource 40103, User 40101 requests the use of System Resource 40103 from Operating System 40102, and Operating System 40102 in turn commands CS 10110 to provide the requested Resources 40103. For example, when a user program wishes to use a peripheral device, it does not deal with the device directly, but instead calls the Operating System 40102 procedure 602 that controls the device. While Operating System 40102 must take into account the device's complicated physical properties, the user program that requested the device need know nothing about the physical properties, but must only know what information the Operating System 40102 Procedure 602 requires to perform the operation requested by the user program. For example, while the peripheral device may require that a precise pattern of data be presented to it, the Operating System 40102 procedure 602 may only require the data itself from the user program, and may format the data as required by the peripheral device. The Operating System 40102 Procedure 602 that controls the peripheral device thus transforms a complicated physical interface to the device into a much simpler logical interface.

1. Resources Controlled by Operating Systems (Fig. 402)

Operating Systems 40102 control two kinds of resources: physical resources and virtual resources. The physical resources in the present embodiment of CS 10110 are JP 10114, IOS 10116 and the peripheral

devices associated with IOS 10116, MEM 10112, and Secondary Storage 10124. Virtual resources are resources that the operating system itself defines for users of CS 10110. As was explained above, in controlling how CS 10110's resources are used, Operating System 40102 defines how CS 10110 appears to the users. Instead of the physical resources controlled by Operating System 40102, the user sees a far simpler set of virtual resources. The logical I/O device interface that Operating System 40102 gives the user of a physical I/O device is such a virtual resource. Often, an Operating System 40102 will define sets of virtual resources and multiplex the physical resources among these virtual resources. For instance, Operating System 40102 may define a set of Virtual Processors 612 that correspond to a smaller group of physical processors, and a set of virtual memories that correspond to a smaller group of physical resources. When a user executes a program, it runs on a Virtual Processor 612 and uses virtual memory. It seems to the user of the virtual processor and the virtual memory that he has sole access to a physical processor and physical memory, but in fact, Operating System 40102 is multiplexing the physical processors and memories among the Virtual Processors 612 and virtual memories.

Operating System 40102, too, uses virtual resources. For instance, the memory management portion of an Operating System 40102 may use I/O devices; when it does so, it uses the virtual I/O devices defined by the portion of the Operating System 40102 that manages the I/O devices. One part of Operating System 40102 may also redefine virtual resources defined by other parts of Operating System 40102. For instance, one part of Operating System 40102 may define a set of primitive virtual I/O devices and another part may use these primitive virtual I/O devices to define a set of high-level user-oriented I/O devices. Operating System 40102 thus turns the physical CS 10110 into a hierarchy of virtual resources. How a user of CS 10110 perceives CS 10110 depends entirely on the level at which he is dealing with the virtual resources.

The entity that uses the resources defined by Operating System 40102 is the process. A Process 610 may be defined as the activity resulting from the execution of a program with its data by a sequential processor. Whenever a user requests the execution of a program on CS 10110, Operating System 40102 creates a Process 610 which then executes the Procedures 602 making up the user's program. In physical terms, a process 610 is a set of data bases in memory that contain the current state of the program execution that the process represents. Operating System 40102 causes Process 610 to execute the program by giving Process 610 access to the virtual resources which it requires to execute the program, by giving the virtual resources access to those parts of Process 610's state which they require to perform their operations, and by giving these virtual resources access to the physical resources. The temporary relationship of one resource to another or of a Process 610 to a resource is called a binding. When a Process 610 has access to a given Virtual Processor 612 and Virtual processor 612 has access to process 610's state, process 610 is bound to Virtual Processor 612, and when Virtual processor 612 has access to JP 10114 and Virtual Processor 612's state is loaded into JP 10114 registers, Virtual processor 612 is bound to JP 10114, and JP 10114 can execute SINs contained in Procedures 602 in the program being executed by Process 610 bound to Virtual Processor 612. Binding and unbinding may occur many times in the course of the execution of a program by a Process 610. For instance, if a Process 610 executes a reference to data and the data is not present in MEM 10112, then Operating System 40102 unbinds Process 610's Virtual Processor 612 from JP 10114 until the data is available in MEM 10112. If the data is not available for an extended period of time, or if the user for whom Process 610 is executing the program wishes to stop the execution of the program for a while, Operating System 40102 may unbind process 610 from its Virtual Processor 612. Virtual Processor 612 is then available for use by other Processes 610.

As mentioned above, the binding process involves giving a first resource access to a second resource, and using the first resource's state in the second resource. To permit binding and unbinding, Operating System 40102 maintains data bases that contain the current state of each resource and each Process 610. State may be defined as the information that the operating system must have to use the resource or execute the Process 610. The state of a line printer, for instance, may be variables that indicate whether the line printer is busy, free, off line, or out of order. A Process 610's state is more involved, since it must contain enough information to allow Operating System 40102 to bind Process 610 to a Virtual Processor 612, execute Process 610 for a while, unbind Process 610, and then rebind it and continue execution where it was halted. A process 610's state thus includes all of the data used by Process 610 up to the time that it was unbound from a Virtual Processor 612, along with information indicating whether Process 610 is ready to begin executing again.

Figure 402 shows the relationship between Processes 610, virtual, and physical resources in an operating system. The figure shows a multi-process Operating System 40102, that is, one that can multiplex CS 10110 resources among several Processes 610. The Processes 610 thus appear to be executing concurrently. The solid arrows in Figure 402 indicate bindings between virtual resources or between virtual and physical resources. Each Process 610 is created by Operating System 40102 to execute a user program. The program consists of Procedures 602, and Process 610 executes Procedures 602 in the order prescribed by the program. Processes 610 are created and managed by a component of Operating System 40102 called the Process Manager. Process Manager 40203 executes a Process 610 by binding it to a Virtual Processor 612. There may be more Processes 610 than there are Virtual Processors 612. In this case, Operating System 40102 multiplexes Virtual Processors 612 among Processes 610.

Virtual Processors 612 are created and made available by another component of Operating System 40102, Virtual Processor Manager 40205. Virtual Processor Manager 40205 also multiplexes JP 10114

among Virtual Processors 612. If a Virtual Processor 612 is ready to run, Virtual Processor Manager 40205 binds it to JP 10114. When Virtual Processor 612 can run no longer, or when another Virtual Processor 612 requires JP 10114, Virtual Processor Manager 40205 unbinds running Virtual Processor 612 from JP 10114 and binds another Virtual Processor 612 to it.

Virtual Processors 612 use virtual memory and I/O resources to perform memory access and input-output. Virtual Memory 40206 is created and managed by Virtual Memory Manager 40207, and Virtual I/O Devices 40208 are created and managed by Virtual I/O Manager 40209. Like Virtual Processor Manager 40205, Components 40207 and 40209 of Operating System 40102 multiplex physical resources among the virtual resources. As described above, one set of virtual resources may use another set. One way in which this can happen is indicated by the broken arrows in Figure 402. These arrows show a binding between Virtual Memory 40206 and Virtual I/O Device 40208. This binding occurs when Virtual Memory 40206 must handle a reference to data contained on a peripheral device such as a disk drive. To the user of Virtual Memory 40206, all data appears to be available in MEM 10110. In fact, however, the data is stored on peripheral devices such as disk drives, and copied into MEM 10112 when required. When a Process 610 references data that has not been copied into MEM 10112, Virtual Memory 40206 must use IOS 10116 to copy the data into MEM 10112. In order to do this, it uses a Virtual I/O Device 40208 provided by Virtual I/O Manager 40209.

b. The Operating System in CS 10110

For the sake of clarity, Operating System 40102 has been described as though it existed outside of CS 10110. In fact, however, Operating System 40102 itself uses the resources it controls. In the present embodiment, parts of Operating System 40102 are embodied in JP 10114 hardware devices, parts are embodied in microcode which executes on JP 10114, and parts are embodied in Procedures 602. These Procedures 602 are sometimes called by Processes 610 executing user programs, and sometimes by special Operating System Processes 610 which do nothing but execute operations for Operating System 40102.

The manner in which the components of Operating System 40102 interact may be illustrated by the way in which CS 10110 handles a page fault, i.e., a reference to data which is not available in MEM 10110. The first indication that there may be a page fault is an ATU Miss Event Signal. This Event Signal is generated by ATU 10228 in FU 10120 when there is no entry in ATU 10228 for a Logical Descriptor 27118 used in a read or write operation. The Event Signal invokes Operating System 40102 microcode, which examines a table in MEM 10112 in order to find whether the data described by Logical Descriptor 27118 has a copy in MEM 10112. If the table indicates that there is no copy, Operating System 40102 microcode communicates the fact of the page fault to an Operating System 40102 Virtual Memory Manager process 610 and removes Virtual Processor 612 bound to the Process 610 which was executing when the page fault occurred from JP 10114. Some time later, Virtual Memory Manager Process 610 is bound to JP 10114. Procedures 602 executed by Virtual Memory Manager Process 610 then initiate the I/O operations required to locate the desired data in Secondary Storage 10124 and copy it into MEM 10112. When the data is available in MEM 10112, Operating System 40102 allows Virtual Processor 612 bound to Process 610 which was executing when the page fault occurred to return to JP 10114. Virtual Processor 612 repeats the memory reference which caused the page fault, and since the data is now in MEM 10112, the reference succeeds and execution of Process 610 continues.

c. Extended Operating System and the Kernel Operating System (Fig. 403)

In CS 10110, Operating System 40102 is made up of two component operating systems, the Extended Operating System (EOS) and the Kernel Operating System (KOS). The KOS has direct access to the physical resources. It defines a set of primitive virtual resources and multiplexes the physical resources among the primitive virtual resources. The EOS has access to the primitive virtual resources defined by KOS, but not to the physical resources. The EOS defines a set of user-level virtual resources and multiplexes the primitive virtual resources defined by KOS among the user level virtual resources. For example, KOS provides EOS with Processes 610 and Virtual processors 612 and binds Virtual Processors 612 to JP 10114, but EOS decides when a Process 610 is to be created and when a process 610 is to be bound to a Virtual processor 612.

Figure 403 shows the relationship between a user Process 610, EOS, KOS, and the physical resources in CS 10110. Figure 403 shows three levels of interface between executing user Process 610 and JP 10114. The highest level of interface is Procedure Level 40302. At this level, Process 610 interacts with CS 10110 by calling Procedures 602 as specified by the program Process 610 is executing. The calls may be either calls to User Procedures 40306 or calls to EOS Procedures 40307. When Process 610 is executing a procedure 602, Process 610 produces a stream of SInS. The stream contains two kinds of SInS, S-language SInS 40310 and KOS SInS 40311. Both kinds of SInS interact with CS 10110 at the next level of interface, SInS-level interface 40309. SInS 40310 and 40311 are interpreted by Microcode 40312 and 40313, and Microinstructions 40315 interact with CS 10110 at the lowest level of interface, JP 10114 interface 40316. As already explained in the discussion of the FU 10120 micromachine, certain conditions in JP 10114 result in

Event Signals 40314 which invoke microroutines in S-Interpreter Microcode 40312 or KOS Microcode 40313. Only Procedure-Level Interface 40302 and SIN-level Interface 40309 are visible to users. Procedure-level Interface 40309 appears as calls in user Procedures 602 or as statements in user Procedures 602 which compilers translate into calls to EOS procedures 602. SIN-level Interface 40309 appears as the Name Tables 10335 and SInS in Procedure Objects 608 generated by compilers.

As Figure 403 indicates, EOS exists only at Procedural Level 40302, while KOS exists at Procedural Level 40302, and SIN Level 40304, and within the microcode beneath SIN Level 40309. The only portion of the operating system that is directly available to user Processes 610 is EOS Procedures 40307. EOS Procedures 40307 may in turn call KOS procedures 40308. In many cases, an EOS Procedure 40307 will contain nothing more than the call to a KOS Procedure 40308.

User Procedures 40306, EOS Procedures 40307, and KOS Procedures 40308 all contain S-language SInS 40310. In addition, KOS Procedures 40308 only may contain special KOS SInS 40311. Special KOS SInS 40311 control functions that are not available to EOS Procedures 40307 or User Procedures 40306, and KOS SInS 40311 may therefore not appear in Procedures 40306 or 40307. S-language SInS 40310 are interpreted by S-Interpreter Microcode 40312, while KOS SInS 40311 are interpreted by KOS Microcode 40313. KOS Microcode 40313 may also be called by S-Interpreter Microcode 40312. Depending on the hardware conditions that cause Event Signals 40314, Signals 40314 may cause the execution of either S-Interpreter Microcode 40312 or KOS Microcode 40313.

Figure 403 shows the system as it is executing a user Process 610. There are in addition special Processes 610 reserved for KOS and EOS use. These Processes 610 work like user Processes 610, but carry out operating system functions such as process management and virtual memory management. With one exception, EOS Processes 610 call EOS Procedures 40307 and KOS Procedures 40308, while KOS Processes 610 call only KOS Procedures 40308. The exception is the beginning of Process 610 execution: KOS performs the KOS-level functions required to begin executing a Process 610 and then calls EOS. EOS performs the required EOS level functions and then calls the first User Procedure 40306 in the program Process 610 is executing.

A description of how KOS handles page faults can serve to show how the parts of the system at the JP 10114, SIN, and procedure Levels work together. A page fault occurs when a Process 610 references a data item that has no copy in MEM 10112. The page fault begins as an Event Signal from ATU 10228. The Event Signal invokes a microroutine in KOS Microcode 40313. If the microroutine confirms that the referenced data item is not in MEM 10112, it records the fact of the page fault in some KOS tables in MEM 10112 and calls another KOS microroutine that unbinds Virtual Processor 612 bound to Process 610 that caused the page fault from JP 10114 and allows another Process 610's Virtual Processor 612 to run. Some time after the page fault, a special operating system Process 610, the Virtual Memory Manager Process 610, runs and executes KOS Procedures 40309. Virtual Memory Manager Process 610 initiates the I/O operation that reads the data from Secondary Storage 10124 into MEM 10112. When IOS 10116 has finished the operation, Process 610 that caused the page fault can run again and Virtual Memory Manager Process 610 performs an operation which causes Process 610's Virtual Processor 612 to again be bound to JP 10114. When Process 610 resumes execution, it again attempts to reference the data. The data is now in MEM 10112 and consequently, the page fault does not recur.

The division of Operating System 40102 into two hierarchically-related operating systems is characteristic for CS 10110. Several advantages are gained by such a division:

- Each of the two operating systems is simpler than a single operating system would be. EOS can concern itself mainly with resource allocation policy and high-level virtual resources, while KOS can concern itself with low-level virtual resources and hardware control.
- Because each operating system is simpler, it is easier to verify that each system's components are performing correctly, and the two systems are therefore more dependable than a single system.
- Dividing Operating System 40102 makes it easier to implement different embodiments of CS 10110. Only the interface provided by EOS is visible to the user, and consequently, the user interface to the system can be changed without altering KOS. In fact, a single CS 10110 may have a number of EOSs, and thereby present different interfaces to different users. Similarly, changes in the hardware affect the implementation of the KOS, but not the interface that KOS provides EOS. A given EOS can therefore run on more than one embodiment of CS 10110.
- A divided operating system is more secure than a single operating system. Physical access to JP 10114 is provided solely by KOS, and consequently, KOS can ensure that users manipulate only those resources to which they have access rights.

All CSs 10110 will have the virtual resources defined by KOS, while the resources defined by EOS will vary from one CS 10110 to another and even within a single CS 10110. Consequently, the remainder of the discussion will concern itself with KOS.

The relationship between the KOS and the rest of CS 10110 is governed by four principles:

- Only the KOS has access to the resources it controls. User calls to EOS may result in EOS calls to KOS, and S-language SInS may result in invocations of KOS microcode routines, but neither EOS nor user programs may directly manipulate resources controlled by KOS.
- The KOS is passive. It responds to calls from the EOS, to microcode invocations, and to Event Signals, but it initiates no action on its own.

— The KOS is invisible to all system users but the EOS. KOS does not affect the logical behavior of a Process 610 and is noticeable to users only with regard to the speed with which a Process 610 executes on CS 10110.

As discussed above, KOS manages both physical and virtual resources. The physical resources and some of the virtual resources are visible only within KOS; others of the virtual resources are provided to EOS. Each virtual resource has two main parts: a set of data bases that contain the virtual resource's state, and a set of routines that manipulate the virtual resource. The set of routines for a virtual resource are termed the resource's manager. The routines may be KOS Procedures 40308, or they may be KOS Microcode 40313. As mentioned, in some cases, KOS uses separate Processes 610 to manage the resources.

For the purposes of this specification, the resources managed by KOS fall into two main groups: those associated with objects, and those associated with Processes 610. In the following, first those resources associated with objects, and then those associated with Processes 610 are discussed.

B. Objects and Object Management (Fig. 404)

The virtual resources termed objects are defined by KOS and manipulated by EOS and KOS. Objects as seen by EOS have five properties:

- A single UID that identifies the object throughout the object's life and specifies what Logical Allocation Unit (LAU) the object belongs to.
- A set of attributes that describe the object and limit access to it.
- Bit-addressable contents. In the present embodiment, the contents may range from 0 to $(2^{**32}) - 1$ bits in length. Any bit in the contents may be addressed by an offset.
- Objects may be created.
- Objects may be destroyed.

All of the data and Procedures 602 in a CS 10110 are contained in objects. Any process 610 executing on a CS 10110 may use a UID-off set address to attempt to access data or Procedures 602 in certain objects on any CS 10110 accessible to the CS 10110 on which Process 610 is executing. The objects which may be thus accessed by any Process 610 are those having UIDs which are guaranteed unique for all present and future CS 10110. Objects with such unique UIDs thus form a single address space which is at least potentially accessible to any process 610 executing on any CS 10110. As will be explained in detail later, whether a Process 610 can in fact access an object in this single address space depends on whether Process 610 has access rights to the object. Other objects, whose UIDs are not unique, may be accessed only by Processes 610 executing on CSs 10110 or groups of CSs 10110 for which the non-unique UID is in fact unique. No two objects accessible to a CS 10110 at a given time may have identical UIDs.

The following discussion of objects will first deal with objects as they are seen directly by EOS and indirectly by user programs, and then deal with objects as they appear to KOS.

Figure 404 illustrates how objects appear to EOS. The object has three parts: the UID 40401, the Attributes 40404, and the Contents, 40406. The object's contents reside in a Logical Allocation Unit (LAU), 40405. UID 40401 has two parts: a LAU Identifier (LAUID) 40402 that indicates what LAU 40405 the object is on, and the Object Serial Number (OSN) 40403, which specifies the object in LAU 40405.

The EOS can create an object on a LAU 40405, and given the object's UID 40401, can destroy the object. In addition, EOS can read and change an object's Attributes 40404. Any Process 610 executing on a CS 10110 may reference information in an object by specifying the object's UID 40401 and the bit in the object at which the information begins. At the highest level, addresses in CS 10110 thus consist of a UID 40401 specifying an object and an offset specifying the number of bits into the object at which the information begins. As will be explained in detail below, KOS translates such UID-offset addresses into intermediate forms called AON-offset addresses for use in JP 10114 and into page number-displacement addresses for use in referencing information which has been copied into MEM 10112.

The physical implementation and manipulation of objects is restricted solely to KOS. For instance, objects and their attributes are in fact stored in Secondary Storage 10124. When a program references a portion of an object, KOS copies that portion of the object from Secondary Storage 10124 into MEM 10112, and if the portion in MEM 10112 is changed, updates the copy of the object in Secondary Storage 10124. EOS and user programs cannot control the location of an object in Secondary Storage 10124 or the location of the copy of a portion of an object in MEM 10112, and therefore can access the object only by means of KOS.

While EOS cannot control the physical implementation of an object, it can provide KOS with information that allows KOS to manage objects more effectively. Such information is termed hints. For instance, KOS generally copies a portion of an object into MEM 10112 only if a Process 610 references information in the object. However, EOS schedules Process 610 execution, and therefore can predict that certain objects will be required in the near future. EOS can pass this information on to KOS, and KOS can use the information to decide what portions of objects to copy into MEM 10112.

a. Objects and User Programs (fig. 405)

As stated above, user programs manipulate objects, but the objects are generally not directly visible to user programs. Instead, user programs use symbols such as variable names or other references to refer to

data stored in objects or file names to refer to the objects themselves. The discussion of Namespace has already illustrated how CS 10110 compilers translate variable names appearing in statements in Procedures 602 into Names, i.e., indexes of NTEs 30401, how Name Resolve microcode resolves NTE 30401 into Logical Descriptors 27116, and how ATU 10228 translates Logical Descriptors 27116 into locations in MEM 10112 containing copies of the portions of the objects in which the data represented by the variables resides.

The translation of filenames to UIDs 40401 is accomplished by EOS. EOS maintains a filename translation table which establishes a relationship between a system filename called a pathname and the UID 40401 of the object containing the file's data, and thereby associates the pathname with the object. A Pathname is a sequence of ASCII characters which identifies a file to a user of CS 10110. Each pathname in a given CS 10110 must be unique. Figure 405 shows the filename translation table. Referring to that figure, when a user gives pathname 40501 to the EOS, EOS uses Filename Translation Table 40503 to translate pathname 40501 into UID 40401 for object 40504 containing the file. An object in CS 10110 may thus be identified in two ways: by means of its UID 40401 or by means of a Pathname 40501. While an object has only a single UID 40401 throughout its life, the object may have many Pathnames 40501. All that is required to change an object's pathname 40501 is the substitution of one Pathname 40501 for another in the object's Entry 40502 in Filename Translation Table 40503. One consequence of the fact that an object may have different Pathnames 40501 during its life is that when a program uses a Pathname 40501 to identify an object, a user of CS 10110 may make the program process a different object simply by giving the object which formerly had Pathname 40501 which appears in the program a new Pathname 40501 and giving the next object to be processed the Pathname 40501 which appears in the program.

In the present embodiment, an object may contain only a single file, and consequently, a Pathname 40501 always refers to an entire object. In other embodiments, a Pathname 40501 may refer to a portion of an object, and in such embodiments, Filename Translation Table 40503 will associate a Pathname 40501 with a UID-offset address specifying the beginning of the file.

b. UIDs 40401 (Fig. 406)

UIDs 40401 may identify objects and other entities in CS 10110. Any entity identified by a UID 40401 has only a single UID throughout its life. Figure 406 is a detailed representation of a CS 10110 UID 40401. UID 40401 is 80 bits long, and has two fields. Field 40402, 32 bits long, is the Logical Allocation Unit Identifier (LAUID). It specifies LAU 40405 containing the object. LAUID 40402 is further subdivided into two subfields: LAU Group Number (LAUGN) 40607 and LAU Serial Number (LAUSN) 40605. LAUGN 40607 specifies a group of LAUs 40405, and LAUSN 40605 specifies a LAU 40405 in that group. Purchasers of CS 10110 may obtain LAUGNs 40607 from the manufacturer. The manufacturer guarantees that he will assign LAUGN 40607 given the purchaser to no other CS 10110, and thus these LAUGNs 40607 may be used to form UIDs 40401 which will be unique for all CSs 10110. Field 40604, 48 bits long, is the Object Serial Number (OSN). It specifies the object in LAU 40405.

UIDs 40401 are generated by KOS Procedures 602.

There are two such procedures 602, one which generates UIDs 40401 which identify objects, and another which generates UIDs 40401 which identify other entities in CS 10110. The former Procedure 602 is called Generate Object UID, and the latter Generate Non-object UID. The Generate Object UID Procedure 602 is called only by the KOS Create Object Procedure 602. Create Object Procedure 602 provides Generate Object UID Procedure 602 with a LAUID 40402, and Generate Object UID Procedure 602 returns a UID 40401 for the object. In the present embodiment, UID 40401 is formed by taking the current value of the architectural clock, contained in a location in MEM 10112, forming an OSN 40403 from the architectural clock's current value, and concatenating OSN 40403 to LAUID 40402.

Generate Non-object UID Procedure 602 may be invoked by EOS to provide a UID 40401 which does not specify an object. Non-object UIDs 40401 may be used in CS 10110 wherever a unique label is required. For example, as will be explained in detail later, all Virtual processors 612 which are available to CS 10110 have non-object UIDs 40401. All such non-object UIDs 40401 have a single LAUSN 40607, and thus, EOS need only provide a LAUGN 40605 as an argument. Generate Non-object UID Procedure 602 concatenates LAUGN 40605 with the special LAUSN 40607, and LAUID 40402 thus produced with an OSN 40403 obtained from the architectural clock. In other embodiments, OSNs 40403 for both object and non-object UIDs 40401 may be generated by other means, such as counters.

CS 10110 also has a special UID 40401 called the Null UID 40401. The Null UID 40401 contains nothing but 0 bits, and is used in situations which require a UID value which cannot represent an entity in CS 10110.

c. Object Attributes

What a program can do with an object is determined by the object's Attributes 40404. There are two kinds of Attributes 40404: Object Attributes and Control Attributes. Object Attributes describe the object's contents; Control Attributes control access to the object. Objects may have Attributes 40404 even though they have no Contents 40406, and in some cases, objects may even exist solely for their Attributes 40404.

For the purposes of this discussion, there are two kinds of Object Attributes: the Size Attribute and the Type Attributes.

An object's Size Attribute indicates the number of bits that the object currently contains. On each

reference to an object's Contents 40406, KOS checks to make sure that the data accessed does not extend beyond the end of the object. If it does, the reference is aborted.

The Type Attributes indicate what kind of information the object contains and how that information may be used. There are three categories of Type Attributes: the Primitive Type Attributes, the Extended Type Attribute, and the Domain of Execution attribute. An object's Primitive Type Attribute indicates whether the object is a data object, a Procedure Object 608, an Extended Type Manager, or an S-interpreter. As their names imply, data objects contain data and Procedure Objects 608 contain Procedures 602. Extended Type Managers (ETMs) are a special type of Procedure Object 608 whose Procedures 608 may perform operations solely on objects called Extended Type Objects. Extended Type Objects (ETOs) are objects which have an Extended Type Attribute in addition to their Primitive Type Attribute; for details, see the discussion of the Extended Type Attribute below. S-interpreters are objects that contain interpreters for S-languages. In the present embodiment, the interpreters consist of dispatch tables and microcode, but in other embodiments, the interpreters may themselves be written in high-level languages. Like the Length Attribute, the Primitive Type Attributes allow KOS to ensure that a program is using an object correctly. For instance, when the KOS executes a call for a Procedure 602 it checks whether the object specified by the call is a Procedure Object 608. If it is not, the call fails.

d. Attributes and Access Control

The remaining Object Attributes and the Control Attributes are all part of CS 10110's Access Control System. The Access Control System is discussed in detail later; here, it is dealt with only to the extent required for the discussion of objects. In CS 10110, an access of an object occurs when a Process 610 fetches SInS contained in a Procedure Object 608, reads data from an object, writes data to an object, or in some cases, when Process 610 transfers control to a Procedure 602. The Access Control System checks whether a Process 610 has the right to perform the access it is attempting. There are two kinds of access in CS 10110, Primitive Access and Extended Access. Primitive Access is access which the Access Control System checks on every reference to an object by a Process 610; Extended Access is access that is checked only on user request. Primitive access checks are performed on every object; extended access checks may be performed only on ETOs, and may be performed only by Procedures 602 contained in ETMs.

The means by which the Access Control System checks a Process 610's access to an object are Process 610's subject and the object's Access Control Lists (ACLs). Each Process 610 has a subject made up of four UIDs 40401. These UIDs 40401 specify the following:

- The user for whom Process 610 was created. This UID 40401 is termed the principal component of the subject.
- Process 610 itself. This UID 40401 is termed the process component.
- The domain in which Process 610 is currently executing. This UID 40401 is termed the domain component.
- A user-defined subgroup of subjects. This UID 40401 is termed the tag component.

A domain is a group of objects which may potentially be accessed by any Process 610 which is executing a Procedure 602 in one of a group of Procedure Objects 608 or ETMs. Each Procedure Object 608 or ETM has a Domain of Execution (DOE) Attribute. This attribute is a UID 40401, and while a Process 610 is executing a Procedure 602 in that Procedure Object 608 or ETM, the DOE attribute UID 40401 is the domain component in Process 610's subject. The DOE attribute thus defines a group of objects which may be accessed by a Process 610 executing Procedures 602 from Procedure Object 608. The group of objects is called Procedure Object 608's domain. As may be seen from the above definition, a subject's domain component may change on any call to or return from a Procedure 602. The tag component may change whenever the user desires. The principal component and the process component, on the other hand, do not change for the life of Process 610.

The ACLs which make up the other half of the Access Control System are attributes of objects. Each ACL consists of a series of Entries (ACLE), and each ACLE has two parts: a Subject Template and a set of Access Privileges. The Subject Template defines a group of subjects, and the set of Access Privileges define the kinds of access that subjects belonging to the group have to the object. To check whether an access to an object is legal, the KOS examines the ACLs. It allows access only if it finds an ACLE whose Subject Template matches the current subject of Process 610 which wishes to make the access and whose set of Access Privileges includes the kind of access desired by Process 610. For example, a Procedure Object 608 may have an ACL with two entries: one whose Subject Template allows any subject access, and whose set of Access Privileges allows only Execute Access, and another whose Subject Template allows only a single subject access and whose set of Access Privileges allows Read, Write, and Execute Access. Such an ACL allows any user of CS 10110 to execute the Procedures 602 in Procedure Object 608, but only a specified Process 610 belonging to a specified user and executing a specified group of Procedures 602 may examine or modify the Procedures 602 in the Procedure Object 608.

There are two kinds of ACLs. All objects have Primitive Access Control Lists (PACLs); ETOs may in addition have Extended Access Control Lists (EACLs). The subject portion of the ACLE is the same in all ACLs; the two kinds of list differ in the kinds of access they control. The access controlled by the PACL is defined by KOS and is checked by KOS on every attempt to gain such access; the access controlled by the EACL is defined by the user and is checked only when the user requests KOS to do so.

e. Implementation of Objects

1. Introduction (Fig. 407, 408)

The user of a CS 10110 need only concern himself with objects as they have just been described. In order for a Process 610 to reference an object, the object's LAU 40405 must be accessible from CS 10110 upon which Process 610 is running, Process 610 must know the object's UID 40401, and Process 610's current subject must have the right to access the object in the desired manner. Process 610 need know neither how the object's Contents 40406 and Attributes 40404 are stored on CS 10110's physical devices nor the methods CS 10110 uses to make the object's Contents 40406 and Attributes 40404 available to Process 610.

The KOS, on the other hand, must implement objects on the physical devices that make up CS 10110. In so doing, it must take into account two sets of physical limitations:

- In logical terms, all CSs 10110 have a single logical memory, but the physical implementation of memory in the system is hierarchical: a given CS 10110 has rapid access to a relatively small MEM 10112, much slower access to a relatively large amount of slow Secondary Storage 10124, and very slow access to LAUs 40405 on other accessible CSs 10110.
- UIDs 40401, and even more, subjects, are too large to be handled efficiently on JP 10114's internal data paths and in JP 10114's registers.

The means by which the KOS overcomes these physical limitations will vary from embodiment to embodiment. Here, there are presented first an overview and then a detailed discussion of the means used in the present embodiment.

The physical limitations of the memory are overcome by means of a Virtual Memory system. The Virtual Memory System creates a one-level logical memory by automatically bringing copies of those portions of objects required by executing Processes 610 into MEM 10112 and automatically copying altered portions of objects from MEM 10112 back to Secondary Storage 10124. Objects thus reside primarily in Secondary Storage 10124, but copies of portions of them are made available in MEM 10112 when a Process 610 makes a reference to them. Besides bringing portions of objects into MEM 10112, when required, the Virtual Memory System keeps track of where in MEM 10112 the portions are located, and when a Process 610 references a portion of an object that is in MEM 10112, the Virtual Memory System translates the reference into a physical location in MEM 10112.

JP 10114's need for smaller object identifiers and subject identifiers is satisfied by the use of internal identifiers called Active Object Numbers (AONs) and Active Subject Numbers (ASNs) inside JP 10114. Each time a UID 40401 is moved from MEM 10112 into JP 10114's registers, it is translated into an AON, and the reverse translation takes place each time an AON is moved from a JP 10114's registers to MEM 10112. Similarly, the current subjects of Processes 610 which are bound to Virtual Processors 612 are translated from four UIDs 40401 into small integer ASNs, and when Virtual Processor 612 is bound to JP 10114, the ASN for the subject belonging to Virtual Processor 612's process 610 is placed in a JP 10114 register. The translations from UID 40401 to AON and vice-versa, and from subject to ASN are performed by KOS.

When KOS translates UIDs 40401 to AONs and vice-versa, it uses AOT 10712. An AOT 10712 Entry (AOTE) for an object contains the object's UID 40401, and the AOTE's index in AOT 10712 is that object's AON. Thus, given an object's AON, KOS can use AOT 10712 to determine the object's UID 40401, and given an object's UID 40401, KOS can use AOT 10712 to determine the object's AON. If the object has not been referenced recently, there may be no AOTE for the object, and thus no AON for the object's UID 40401. Objects that have no AONs are called inactive objects. If an attempt to convert a UID 40401 to an AON reveals that the object is inactive, an Inactive Object Fault results and KOS must activate the object, that is, it must assign the object an AON and make an AOTE for it.

KOS uses AST 10914 to translate subjects into ASN's. When a Process 610's subject changes, AST 10914 provides Process 610 with the new subject's ASN. A subject may presently have no ASN associated with it. Such subjects are termed inactive subjects. If a subject is inactive, an attempt to translate the subject to an ASN causes KOS to activate the subject, that is, to assign the subject an ASN and make an entry for the subject in AST 10914.

In order to achieve efficient execution of programs by Processes 610, KOS accelerates information that is frequently used by executing processes 610. There are two stages of acceleration:

- Tables that contain the information are wired into MEM 10112, that is, the Virtual Memory System never uses MEM 10112 space reserved for the tables for other purposes.
- Special hardware devices in JP 10114 contain portions of the information in the tables.

MHT 10716, AOT 10712, and AST 10914 are examples of the first stage of acceleration. As previously mentioned, these tables are always present in MEM 10112. Address Translation Unit (ATU) 10228 is an example of the second stage. As previously explained, ATU 10228 is a hardware cache that contains copies of the most recently used MHT 10716 entries. Like MHT 10716, it translates AON offset addresses into the MEM 10112 locations that contain copies of the data that the UID-offset address corresponding to the AON-offset address refers to. ATU 10228 is maintained by KOS Logical Address Translation (LAT) microcode.

Figure 407 shows the relationship between ATU 10228, MEM 10112, MHT 10716, and KOS LAT microcode 40704. When JP 10114 makes a memory reference, it passes AON-offset Address 40705 to ATU 10228. If ATU 10228 contains a copy of MHT 10716's entry for Address 40705, it immediately produces the corresponding MEM 10112 Address 40706 and transmits the address to MEM 10112. If there is no copy,

ATU 10228 produces an ATU Miss Event Signal which invokes LAT microcode 40704 in JP 10114. LAT microcode 40704 obtains the MHT entry that corresponds to the AON-offset address from MHT 10716, places the entry in ATU 10228, and returns. JP 10114 then repeats the reference. This time, there is an entry for the reference, and ATU 10228 translates the AON address into the address of the copy of the data contained in MEM 10112.

The relationship between KOS table, hardware cache, and microcode just described is typical for the present embodiment of CS 10110. The table (in this case, MHT 10716), is the primary source of information and is maintained by the Virtual Memory Manager Process, while the cache accelerates portions of the table and is maintained by KOS microcode that is invoked by event signals from the cache.

AOT 10712, AST 10914, and MHT 10716 share another characteristic that is typical of the present embodiment of CS 10110: the tables are constructed in such a fashion that the table entry that performs the desired translation is located by means of a hash function and a hash table. The hash function translates the large UID 40401, subject, or AON into a small integer. This integer is the index of an entry in the hash table. The contents of the hash table entry is an index into AOT 10712, AST 10914, or MHT 10716, as the case may be, and these tables are maintained in such a fashion that the entry corresponding to the index provided by the hash table is either the entry that can perform the desired translation or contains information that allows KOS to find the desired entry. The entries in the tables furthermore contain the values they translate. Consequently, KOS can hash the value, find the entry, and then check whether the entry is the one for the hashed value. If it is not, KOS can quickly go from the entry located by the hash table to the correct entry.

Figure 408 shows how hashing works in AST 10914 in the present embodiment. In the present embodiment, Subject 40801, i.e., the principal, process, and domain components of the current subject, are input into Hash Function 40802. Hash Function 40802 produces the index of an entry in ASTHT 10710. ASTHT Entry 40504 in turn contains the index of an Entry (ASTE) 40806 in AST 10914. These 40806 indexes are ASNs. ASTE 40806 contains the principal, process, and domain components of some subject and a link field pointing to ASTE 40806'. ASTE 40806' has 0 in its link field, which indicates that it is the last link in the chain of ASTES beginning with ASTE 40806. If the hashing of a subject yields ASTE 40806, KOS compares the subject in ASTE 40806 with the hashed subject; if they are identical, ASTE 40806's index in AST 10914 is the subject's ASN. If they are not identical, KOS uses the link in ASTE 40806 to find ASTE 40806'. It compares the subject in ASTE 40806' with the hashed subject; if they are identical, ASTE 40806's AST index is the subject's ASN; otherwise, ASTE 40806' is the last entry in the chain, and consequently, there is no ASTE 40806 and no ASN for the hashed subject.

In the following, we will discuss the implementation of objects in the present embodiment in detail, beginning with the implementation of objects in Secondary Storage 10124 and proceeding then to CS 10110's Active Object Management System, the Access Control System, and the Virtual Memory System.

2. Objects in Secondary Storage 10124 (Figs. 409, 410)

As described above, objects are collected into LAUs 40405. The objects belonging to a LAU 40405 are stored in Secondary Storage 10124. Each LAU 40405 contains an object whose contents are a table called the Logical Allocation Unit Directory (LAUD). As its name implies, the LAUD is a directory of the objects in LAU 40405. Each object in LAU 40405, including the object containing the LAUD, has an entry in the LAUD. Figure 409 shows the relationship between Secondary Storage 10124, LAU 40405, the LAUD, and objects. LAU 40405 resides on a number of Storage Devices 40904. LAUD Object 40902' in LAU 40405 contains LAUD 40903. Two LAUDEs 40906 are shown. One contains the attributes of LAUD Object 40902 and the location of its contents, and the other contains the attributes of LAUD Object 40902' containing LAUD 40903 and the location of its contents.

KOS uses a table called the Active LAU Table (ALAUT) to locate the LAUD belonging to LAU 40405. Figure 410 illustrates the relationship between ALAUT 41001, ALAUT Entries 41002, LAUs 40405, and LAUD Objects 40902'. Each LAU 40405 accessible to CS 10110 has an Entry (ALAUITE) 41002 in ALAUT 41001. ALAUITE 41002 for LAU 40405 includes LAU 40405's LAUID 40402 and UID 40401 of LAU 40705's LAUD Object 40902'. Hence, given an object's UID 40401, KOS can use UID 40401's LAUID 40402 to locate ALAUITE 41002 for the object's LAU 40405, and can use ALAUITE 41002 to locate LAU 40405's LAUD 40903. Once LAUD 40903 has been found, OSN portion 40402 of the object's UID 40401 provides the proper LAUDE 40906, and LAUDE 40906 contains object's attributes and the location of its contents.

LAUD 40903 and the Procedures 602 that manipulate it belong to a part of KOS termed the Inactive Object Manager. The following discussion of the Inactive Object Manager will begin with the manner in which an object's contents are represented on Secondary Storage 10124, will then discuss LAUD 40903 in detail, and conclude by discussing the operations performed by Inactive Object Manager Procedures 602.

a.a. Representation of an Object's Contents on Secondary Storage 10124

In general, the manner in which an object's contents are represented on Secondary Storage 10124 depends completely on the Secondary Storage 10124. If a LAU 40405 is made up of disks, then the object's contents will be stored in disk blocks. As long as KOS can locate the object's contents, it makes no difference whether the storage is contiguous or non-contiguous.

In the present embodiment, the objects' contents are stored in files created by the Data General

Advance Operating System (AOS) procedures executing on IOS 10116 These procedures manage files that contain objects' contents for KOS. In future CSs 10110, the representation of an object's contents on Secondary Storage 10124 will be managed by a portion of KOS.

5 b.b. LAUD 40903 (fig. 411, 412)

Figure 411 is a conceptual illustration of LAUD 40903. LAUD 40903 has three parts: LAUD Header 41102, Master Directory 41105, and LAUD Entries (LAUDEs) 40906. LAUD Header 41102 and Master Directory 41105 occupy fixed locations in LAUD 40903, and can therefore always be located from the UID 40401 of LAUD 40903 given in ALAUT 41001. The locations of LAUDEs 40906 are not fixed, but the entry for
10 an individual object can be located from Master Directory 41105.

Turning first to LAUD Header 41102, LAUD Header 41102 contains LAUID 40402 belonging to LAU 40405 to which LAUD 40903 belongs and OSN 40403 of LAUD 40903. As will be explained in greater detail below, KOS can use OSN 40403 to find LAUDE 40906 for LAUD 40903.

Turning now to Master Directory 41105, Master Directory 41105 translates an object's OSN 40403 into
15 the location of the object's LAUDE 40906. Master Directory 41105 contains one Entry 41108 for each object in LAU 40505. Each Entry has two fields: OSN Field 41106 and Offset Field 41107. OSN Field 41106 contains OSN 40403 for the object to which Entry 41108 belongs; Offset Field 41107 contains the offset of the object's LAUDE 40906 in LAUD 40903. KOS orders Entries 41108 by increasing OSN 40403, and can therefore use binary search means to find Entry 41108 containing a given OSN 40403. Once Entry 41108 has
20 been located, Entry 41108's Offset Field 41107, combined with LAUD 40903's OSN 40403, yields the UID offset address of the object's LAUDE 40906.

Once KOS knows the location of LAUDE 40906 it can determine an object's Attributes 40404 and the location of its Contents 40406. Figure 411 gives only an overview of LAUDE 40906's general structure. LAUDE 40906 has three components: a group of fields of fixed size 41109 that are present in every LAUDE
25 40906, and two variable sized components, one, 41139, containing entries belonging to the object's PACL and another, 41141, containing the object's EACL.

As the preceding descriptions of the LAUD's components imply, the number of LAUDEs 40906 and Master Directory Entries 41108 varies with the number of objects in LAU 40405. Furthermore, the amount of space required for an object's EACL and PACL varies from object to object. KOS deals with this problem by
30 including Free Space 41123 in each LAUD 40903. When an object is created, or when an object's ACLs are expanded, the Inactive Object Manager expands LAUD 40903 only if there is no available Free Space 41123; if there is Free Space 41123, the Inactive Object Manager takes the necessary space from Free Space 41123; when an object is deleted or an object's ACLs shortened, the Inactive Object Manager returns the unneeded space to Free Space 41123.

Figure 412 is a detailed representation of a single LAUDE 40906. Figure 412 presents those fields of
35 LAUDE 40906 which are common to all embodiments of CS 10110; fields which may vary from embodiment to embodiment are ignored. Starting at the top of Figure 412, Structure Version Field 41209 contains information by which KOS can determine which version of LAUDE 40906 it is dealing with. Size Field 41211 contains the Size Attribute of the object to which LAUDE 40906 belongs. The Size Attribute specifies the number of bits currently contained in the object. Lock Field 41213 is a KOS lock. As will be explained in detail in the discussion of Processes 610, Lock Field 41213 allows only one Process 610 to read or write LAUDE 40906 at a time, and therefore keeps one Process 610 from altering LAUDE 40906 while
40 another Process 610 is reading LAUDE 40906. File Identifier 41215 contains a system identifier for the file which contains the Contents 40406 of the object to which LAUDE 40906 belongs. The form of File Identifier 41215 may vary from embodiment to embodiment; in the present embodiment, it is an AOS system file identifier. UID Field 41217 contains UID 40401 belonging to LAUDE 40906's object. Primitive Type Field 41219 contains a value which specifies the object's Primitive Type. The object may be a data object, a Procedure Object 608, an ETM, or an S-interpreter object. AON Field 41221 contains a valid value only when LAUDE 40906's object is active, i.e., has an entry in AOT 10712. AON Field 41221 then contains the object's
45 AON. If the object is an ETO, Extended Type Attribute Field 41223 contains the UID 40401 of the ETO's ETM. Otherwise, it contains a Null UID 40401. Similarly, if the object is a Procedure Object 608 or an ETM, Domain of Execution Attribute Field 41225 contains the object's Domain of Execution Attribute.

The remaining parts of LAUDE 40906 belong to the Access Control System and will be explained in detail in that discussion. Attribute Version Number Field 41227 contains a value indicating which version of
50 ACLEs this LAUDE 40906 contains, PACL Size Field 41229 and EACL Size Field 41231 contain the sizes of the respective ACLs, PACL Offset Field 41233 and EACL Offset Field 41235 contain the offsets in LAUD 40903 of additional PACLEs 41139 and EACLEs 41141, and fixed PACLEs 41237 contains the portion of the PACL which is always included in LAUDE 40906.

60 3. Active Objects (fig. 413)

An active object is an object whose UID 40401 has an AON associated with it. In the present
embodiment, each CS 10110 has a set of AONs' KOS associates these AONs with UIDs 40401 in such fashion that at any given moment, an AON in a CS 10110 represents a single UID 40401. Inside FU 10120, AONs are used to represent UIDs CS 10110. In the present embodiment, the AON is represented by 14 bits.
65 A 112-bit UID-offset address (80 bits for UID 40401 and 32 for the offset) is thus represented inside FU 10120

by a 46-bit AON-offset address (14 bits for the AON and 32 bits for the offset).

A CS 10110 has far fewer AONs than there are UIDs 40401. KOS multiplexes a CS 10110's AONs among those objects that are being referenced by CS 10110 and therefore require AONs as well as UIDs 40401. While a given AON represents only a single UID 40401 at any given time, at different times, a UID 40401 may have different AONs associated with it.

Figure 413 provides a conceptual representation of the relationship between AONs and UIDs 40401. Each CS 10110 has potential access to $2^{*}80$ UIDs 40401. Some of these UIDs, however, represent entities other than objects, and others are never associated with any entity. Each CS 10110 also has a set of AONs 41303 available to it. In the present embodiment, this set may have up to $2^{*}14$ values. Since the AONS are only used internally, each CS 10110 may have the same set of AONs 41303. Any AON 41304 in set of AONs 41303 may be associated with a single UID 40401 in set of object UIDs 41301. At different times, an AON 41304 may be associated with different UIDs 40401.

As mentioned above, KOS associates AONs 41304 with UIDs 40401. It does so by means of AOT 10712. Each AOT entry (AOTE) 41306 in AOT 10712 associates a UID 40401 with an AON 41304. AON 41304 is the index of AOTE 41306 which contains UID 40401. Until AOTE 41306 is changed, the AON 41304 which is the index of AOTE 41306 containing UID 40401 represents UID 40401. AOT 10712 also allows UIDs 40401 to be translated into AONs 41303 and vice-versa. Figure 413 illustrates the process for UID-offset Address 41308 and AON-offset Address 41309. AOTE 41306 associates AON 41304 in AON-offset Address 41309 with UID 40401 in UID-offset Address 41308, and Addresses 41308 and 41309 have the same Offset 41307. Consequently, AON-offset Address 41309 represents UID-offset Address 41308 inside JP 10114. Since both addresses use the same Offset, Address 41309 can be translated into address 41308 by translating Address 41309's AON 41304 into Address 41308's UID 40401, and Address 41308 can be translated into Address 41309 by the reverse process. In both cases, the translation is performed by finding the proper AOTE 41306.

The process by which an object becomes active is called object activation. A UID-offset Address 41308 cannot be translated into an AON-offset Address 41309 unless the object to which UID 40401 of UID-offset Address 41308 belongs is active. If a Process 610 attempts to perform such a translation using a UID 40401 belonging to an inactive object, an Inactive Object Fault occurs. KOS handles the fault by removing Process 610 that attempted the translation from JP 10114 until a special KOS Process called the Object Manager Process has activated the object. After the object has been activated, Process 610 may return to JP 10114 and complete the UID 40401 to AON 41304 translation.

The portion of KOS that manages active objects is called the Active Object Manager (AOM). Parts of the AOM are Procedures 602, and parts of it are microcode routines. The high-level language components of the AOM may be invoked only by KOS processes 610. KOS Active Object Manager Process 610 performs most of the functions involved in active object management.

a.a. UID 40401 to AON 41304 Translation

Generally speaking, in CS 10110, addresses stored in MEM 10112 and Secondary Memory 10124 are stored as UID offset addresses. The only form of address that FU 10120 can translate into a location in MEM 10112 is the AON-offset form. Consequently, each time an address is loaded from MEM 10112 into a FU 10120 register, the address must be translated from a UID-offset address to an AON-offset address. The reverse translation must be performed each time an address is moved from a FU 10120 register back into memory.

Such translations may occur at any time. For example, a running Virtual Processor 612 performs such a translation when the Process 610 being executed by Virtual Processor 612 carries out an indirect memory reference. An indirect memory reference is a reference which first fetches a pointer, that is, a data item whose value is the address of another data item, and then uses the address contained in the pointer to fetch the data itself. In CS 10110, pointers represent UID-offset addresses. Virtual Processor 612 performs the indirect memory reference by fetching the pointer from MEM 10112, placing it in FU 10120 registers, translating UID 40401 represented by the pointer into AON 41304 associated with it, and using the resulting AON-offset address to access the data at the location specified by the address.

Most such translations, however, occur when Virtual Processor 612 state is saved or restored. For instance, when one Process 610's Virtual Processor 612 is removed from JP 10114 and another Process 610's Virtual Processor 612 is bound to JP 10114, the state of Virtual Processor 612 being removed from JP 10114 is stored in memory, and the state of Virtual Processor 612 being bound to JP 10114 is moved into JP 10114's registers. Because only UID-offset addresses may be stored in memory, all of the AON-offset addresses in the state of Virtual Processor 612 which is being removed from JP 10114 must be translated into UID-offset addresses. Similarly, all of the UID-offset addresses in the state of Virtual Processor 612 being bound to JP 10114 must be translated into AON-offset addresses before they can be loaded into FU 10120 registers.

C. The Access Control System

As mentioned in the introduction to objects, each time a process 610 accesses data or SINS in an object, the KOS Access Control System checks whether Process 610's current subject has the right to perform the kind of access that Process 610 is attempting. If Process 610's current subject does not have the proper access, the Access Control System aborts the memory operation which Process 610 was attempting to

carry out. The following discussion presents details of the implementation of the Access Control System, beginning with subjects, then proceeding to subject templates, and finally to the means used by KOS to accelerate access checking.

5 a. Subjects

A Process 610's subject is part of process 610's state and is contained along with other state belonging to Process 610 in an object called a Process Object. Process Objects are dealt with at length in the detailed discussion of Processes 610 which follows the discussion of objects. While a subject has, as mentioned above, four components, the principal component, the process component, the domain component, and 10 the tag component, the Access Control System in the present embodiment of CS 10110 assigns values to only the first three components and ignores the tag component when checking access.

In the present embodiment, the UIDs 40401 which make up the components of a Process 610's subject are the UIDs 40401 of objects containing information about the entities represented by the UIDs 40401. The principal component's UID 40401 represents an object called the Principal Object. The Principal Object 15 contains information about the user for whom Process 610 was created. For example, the information might concern what access rights the user had to the resources of CS 10110, or it might contain records of his use of CS 10110. The process component's UID 40401 represents the Process Object, while the domain component's UID 40401 represents an object called the Domain Object. The Domain Object contains information which must be accessible to any Process 610 whose subject has the Domain Object's UID 20 40401 as its domain component. Other embodiments of CS 10110 will use the tag component of the subject. In these embodiments, the tag component's UID 40401 is the UID 40401 of a Tag Object containing at least such information as a list of the subjects which make up the group of subjects represented by the tag component's UID.

25 b. Domains

As stated above, the subject's domain component is the domain of execution attribute belonging to the Procedure Object 608 or ETM whose code is being executed when the access request is made. The domain component of the subject thus gives Process 610 to which the subject belongs potential access to the group of objects whose ACLs have ACLEs with subject templates containing domain components that match the 30 DOE attribute. This group of objects is the domain defined by the Procedure Object 608 or ETM's DOE attribute. When a Process 610 executes a Procedure 602 from a Procedure Object 608 or ETM with a given DOE attribute, Process 610 is said to be executing in the domain defined by that DOE attribute. As may be inferred from the above, different Procedure Objects 608 or ETMs may have the same DOE attribute, and objects may have ACLEs which make them members of many different domains.

In establishing a relationship between a group of Procedure Objects 608 and another group of objects, a domain allows a programmer using CS 10110 to ensure that a given object is read, executed, or modified only by a certain set of Procedures 602. Domains may thus be used to construct protected subsystems in CS 10110. One example of such a protected subsystem is KOS itself: the objects in CS 10110 which contain KOS tables all have ACLs whose domain template components match only the DOE which represents the 40 KOS domain. The only Procedure Objects 608 and ETMs which have this DOE are those which contain KOS Procedures 602, and consequently, only KOS Procedures 602 may manipulate KOS tables.

Since an object may belong to more than one domain, a programmer may use domains to establish hierarchies of access. For example, if some of the objects in a first domain belong both to the first domain and a second domain, and the second domain's objects all also belong to the first domain, then Procedures 45 602 contained in Procedure Objects 608 whose DOEs define the first domain may access any object in the first domain, including those which also belong to the second domain, while those from Procedure Objects 608 whose DOEs define the second domain may access only those objects in the second domain.

c. Access Control Lists

50 As previously mentioned, the Access Control System compares the subject belonging to Process 610 making an access to an object and the kind of access Process 610 desires to make with the object's ACLs to determine whether the access is legal. The following discussion of the ACLs will first deal with Subject Templates, since they are common to all ACLs, and then with PACLs and EACLs.

55 1. Subject Templates (Fig. 416)

Figure 416 shows Subject Templates, PACL Entries (PACLEs), and EACL Entries (EACLEs). Turning first to the Subject Templates, Subject Template 41601 consists of four components, Principal Template 41606, Process Template 41607, Domain Template 41609, and Tag Template 41611. Each template has two fields, Flavor Field 41603, and UID Field 41605. Flavor Field 41603 indicates the way in which the template to which 60 it belongs is to match the corresponding component of the subject for Process 610 attempting the access. Flavor Field 41603 may have one of three values: match any, match one, match group. If Flavor Field 41603 has the value match any, any subject component UID 40401 matches the template, and the Access Control System does not examine UID Field 41605. If Flavor Field 41603 has the value match one, then the corresponding subject component must have the same UID 40401 as the one contained in UID Field 41605. 65 If Flavor Field 41603 has the value match group, finally, then UID Field 41605 contains a UID 40401 of an

object containing information about the group of subject components which the given subject component may match.

2. Primitive Access Control Lists (PACLs)

PACLs are made up of PACLEs 41613 as illustrated in Figure 416. Each PACLE 41613 has two parts: a subject template 41601 and an Access Mode Bits Field 41615. The values in Access Mode Bits Field 41615 define 11 kinds of access. The eleven kinds fall into two groups: Primitive Data Access and Primitive Non-data Access. Primitive Data Access controls what the subject may do with the object's Contents 40406; Primitive Non-data Access controls what the subject may do with the object's Attributes 40404.

There are three kinds of Primitive Data Access: Read Access, Write Access, and Execute Access. If a subject has Read Access, it can examine the data contained in the object; if the subject has Write Access, it can alter the data contained in the object; if it has Execute Access, it can treat the data in the object as a Procedure 602 and attempt to execute it. A subject may have none of these kinds of access, or any combination of the kinds. On every reference to an object, the KOS checks whether the subject performing the reference has the required Primitive Data Access.

Primitive Non-data Access to an object is required only to set or read an object's Attributes 40404, and is checked only when these operations are performed. The kinds of Non-data Access correspond to the kinds of Attributes 40404:

Attributes	Kind of Access
Object Attributes	get object attributes set object attributes
Primitive Control Attributes	get primitive control attributes set primitive control attributes
Extended Control Attributes	get extended control attributes set extended control attributes
ETM Access	use as ETM create ETO

The access rights for object attributes allow a subject to get and set the object attributes described previously. The access rights for primitive and extended control attributes allow a subject to get and set an object's PACL and EACL respectively.

An object may have any number of PACLEs 41613 in its PACL. The first five PACLEs 41613 in an object's PACL are contained in fixed PACLE Field 41237 of LAUDE 40906 for the object; the remainder are stored in LAUD 40903 at the location specified in PACL Offset Field 41233 of LAUDE 40906.

3. APAM 10918 and Protection Cache 10234 (Fig. 421)

Primitive non-data access rights are checked only when users invoke KOS routines that require such access rights, and extended access rights are checked only when users request such checks. Primitive data access rights, on the other hand, are checked every time a Virtual Processor 612 makes a memory reference while executing a Process 610. The KOS implementation of primitive data access right checking therefore emphasizes speed and efficiency. There are two parts to the implementation: APAM 10918 in MEM 10112, and Protection Cache 10234 in JP 10114. APAM 10918 is in a location in MEM 10112 known to KOS microcode. APAM 10918 contains primitive data access information copied from PACLEs 41613 which belong to active objects and whose Subject Template 41601 matches an active subject. Protection Cache 10234, in turn, contain copies of the information in APAM 10918 for the active subject of Process 610 whose Virtual Processor 612 is currently bound to JP 10114 and active objects referenced by Process 610. A primitive data access check in CS 10110 begins with Protection Cache 10234, and if the information is not contained in Protection Cache 10234, proceeds to APAM 10918, and if it is not there, finally, to the object's PACL. The discussion which follows begins with APAM 10918.

Figure 421 shows APAM 10918. APAM 10918 is organized as a two-dimensional array. The array's row indexes are AONs 41304, and its column indexes are ASNs. There is a row for each AON 41304 in CS 10110, and a column for each ASN. In Figure 421, only a single row and column are shown. Any primitive data access information in APAM 10918 for the object represented by AON 41304 j is contained in Row 42104, while Column 42105 contains any primitive data access information in APAM 10918 for the subject

represented by ASN k. APAM Entry (APAME) 42106 is at the intersection of Row 42104 and Column 42105, and thus contains the primitive data access information from that PACLE 41613 belonging to the object represented by AON 41304 j whose Subject Template 41601 matches the subject represented by ASN k.

An expanded view of APAME 42106 is presented beneath the representation of APAM 10918. APAME 42106 contains four 1-bit fields. The bits represent the kinds of primitive data access that the subject represented by APAME 42106's column index has to the object represented by APAME 42106's row index.

— Field 42107 is the Valid Bit. If the Valid Bit is set, APAME 42106 contains whatever primitive data access information is available for the subject represented by the column and the object represented by the row. The remaining fields in APAME 42106 are meaningful only if Valid Bit 42107 is set.

— Field 42109 is the Execute Bit. If it is set, APAME 42106's subject has Execute Access to APAME 42106's object.

— Field 42111 is the Read Bit. If it is set, APAME 42106's subject has Read Access to APAME 42106's object.

— Field 42113 is the Write Bit. If it is set, APAME 42106's subject has Write Access to APAME 42106's object.

Any combination of bits in Fields 42109 through 42113 may be set. If all of these fields are set to 0, APAME 42106 indicates that the subject it represents has no access to the object it represents.

KOS sets APAME 42106 for an ASN and an AON 41304 the first time the subject represented by the ASN references the object represented by AON 41304. Until APAME 42106 is set, Valid Bit 42107 is set to 0. When APAME 42106 is set, Valid Bit 42107 is set to 1 and Fields 42109 through 42113 are set according to the primitive data access information in the object's PACLE 41613 whose Subject Template 41601 matches the subject. When an object is deactivated, Valid Bits 42107 in all APAMEs 42106 in the row belonging to the object's AON 41304 are set to 0; similarly, when a subject is deactivated, Valid Bits 42107 in all APAMEs 42106 in the column belonging to the subject's ASN are set to 0.

4. Protection Cache 10234 and Protection Checking (Fig. 422)

The final stage in the acceleration of protection information is Protection Cache 10234 in JP 10114. The details of the way in which Protection Cache 10234 functions are presented in the discussion of the hardware; here, there are discussed the manner in which Protection Cache 10234 performs access checks, the relationship between protection Cache 10234, APAM 10918, and AOT 10712, and the manner in which KOS protection cache microcode maintains Protection Cache 10234.

Figure 422 is a block diagram of Protection Cache 10234, AOTE 10712, APAM 10918, and KOS Microcode 42207 which maintains Protection Cache 10234. Each time JP 10114 makes a memory reference using a Logical Descriptor 27116, it simultaneously presents Logical Descriptor 27116 and a Signal 42208 indicating the kind of memory operation to Protection Cache 10234 and ATU 10228. Entries 42215 in Protection Cache 10234 contain primitive data access and length information for objects previously referenced by the current subject of Process 610 whose Virtual Processor 612 is currently bound to JP 10114. On every memory reference, Protection Cache 10234 emits a Valid/invalid Signal 42205 to MEM 10112. If Protection Cache 10234 contains no Entry 42215 for AON 41304 contained in Logical Descriptor 27116's AON field 27111, if Entry 42215 indicates that the subject does not have the type of access required by process 610, or if the sum of Logical Descriptor 27116's OFF field 27113 and LEN field 27115 exceed the object's current size, Protection Cache 10234 emits an Invalid Signal 42205. This signal causes MEM 10112 to abort the memory reference. Otherwise, Protection Cache 10234 emits a Valid Signal 42205 and MEM 10112 executes the memory reference.

When Protection Cache 10234 emits an Invalid Signal 42205, it latches Logical Descriptor 27116 used to make the reference into Descriptor Trap 20256, the memory command into Command Trap 27018, and if it was a write operation, the data into Data Trap 20258, and at the same time emits one of two Event Signals to KOS microcode. Illegal Access Event Signal 42208 occurs when Process 610 making the reference does not have the proper access rights or the data referenced extends beyond the end of the object. Illegal Access Event Signal 42208 invokes KOS microcode 42215 which performs a Microcode to Software Call 42217 (described in the discussion of Calls) to KOS Access Control System Procedures 602 and passes the contents of Descriptor Trap 20256, Command Trap 27018, the ASN of Process 610 (contained in a register MGR's 10360), and if necessary, Data Trap 20258 to these Procedures 602. These procedures 602 inform EOS of the protection violation, and EOS can then remedy it.

Cache Miss Event Signal 42206 occurs when there is no Entry 42215 for AON 41304 in protection Cache 10234. Cache Miss Event Signal 42206 invokes KOS Protection Cache Miss Microcode 42207, which constructs missing Protection Cache Entry 42215 from information obtained from AOT 10712 and APAM 10918. If APAM 10918 contains no entry for the current subject's ASN and the AON of the object being referenced, protection Cache Miss Microcode 42207 performs a Microcode-to-software Call to KOS Access Control System Procedures 602 which go to LAUDE 40906 for the object and copy the required primitive data access information from the PACLE 41613 belonging to the object whose Subject Template 41601 matches the subject attempting the reference into APAM 10918. The KOS Access Control System Procedures 602 then return to Cache Miss Microcode 42207, which itself returns. Since Cache Miss Microcode 41107 was invoked by an Event Signal, the return causes JP 10114 to reexecute the memory reference which caused the protection cache miss. If protection Cache 10234 was loaded as a result of the

last protection cache miss, the miss does not recur; if Protection Cache 10234 was not loaded because the required information was not in APAM 10918, the miss recurs, but since the information was placed in APAM 10918 as a result of the previous miss, Cache Miss Microcode 42207 can now construct an Entry 42215 in Protection Cache 10234. When Cache Miss Microcode 42207 returns, the memory reference is again attempted, but this time Protection Cache 10234 contains the information and the miss does not recur.

Cache Miss Microcode 42207 creates a new Protection Cache Entry 42215 and loads it into Protection Cache 10234 as follows: Using AON 41304 from Logical Descriptor 27116 latched into Descriptor Trap 20256 when the memory reference which caused the miss was executed and the current subject's ASN, contained in GR's 10360, Cache Miss Microcode locates APAME 42106 for the subject represented by the ASN and the object represented by AON 41304 and copies the contents of APAME 42106 into a JP 10114 register which may serve as a source for JPD Bus 10142. It also uses AON 41304 to locate AOTE 41306 for the object and copies the contents of Size Field 41519 into another JP 10114 register which is a source for JPD Bus 10142. It then uses three special microcommands, executed in successive microinstructions, to load Protection Cache Entry 42215. The first microcommand loads Protection Cache Entry 42215's TS 24010 with AON 41304 of Logical Descriptor 27116 latched into Descriptor Trap 20256; the second loads the object's size into Protection Cache 10234's EXTENT field, and the third loads the contents of APAME 42106 in the same fashion.

Another microcommand invalidates all Entries 42215 in Protection Cache 10234. This operation, called flushing, is performed when an object is deactivated or when the current subject changes. The current subject changes whenever a Virtual Processor 612 is unbound from JP 10114, and whenever a Process 610 performs a call to or a return from a Procedure 602 executing in a domain different from that in which the calling Procedure 602 or the Procedure 602 being returned to executes in. In the cases of the Call and the unbinding of Virtual Processor 612, the cache flush is performed by KOS Call and dispatching microcode; in the case of object deactivation, it is performed by a KOS procedure using a special KOS SIN which invokes Cache Flush Microcode.

D. Processes

1. Synchronization of Processes 610 and Virtual Processors 612

Since Processes 610 and the Virtual Processors 612 to which they are bound may execute concurrently on CS 10110, KOS must provide means for synchronizing Processes 610 which depend on each other. For example, if process 610 A cannot proceed until Process 610 B has performed some operation, there must be a mechanism for suspending A's execution until B is finished. Generally speaking, four kinds of synchronization are necessary:

- One Process 610 must be able to halt and wait for another Process 610 to finish a task before it proceeds.
- One Process 610 must be able to send another Process 610 a message and wait for a reply before it proceeds.
- When processes 610 share a data base, one Process 610 must be able to exclude other Processes 610 from the data base until the first Process 610 is finished using the data base.
- One Process 610 must be able to interrupt another Process 610, i.e., asynchronously cause the second Process 610 to perform some action.

KOS has internal mechanisms for each kind of synchronization, and in addition supplies synchronization mechanisms to EOS. KOS uses the internal mechanisms to synchronize Virtual Processors 612 and KOS Processes 610, while EOS uses the mechanisms supplied by KOS to synchronize all other Processes 610. The internal mechanisms are the following:

- Event counters, Await Entries, and Await Tables. As will be explained in detail below, Event Counters and Await Entries allow one Process 610 to halt and wait for another Process 610 to complete an operation. Event counters and Await Entries are also used to implement process interrupts. Await Entries are organized into Await Tables.
- Message Queues. Message Queues allow one Process 610 to send a message to another and wait for a reply. Message Queues are implemented with Event Counters and queue data structures.
- Locks. Locks allow one Process 610 to exclude other Processes 610 from a data base or a segment of code. Locks are implemented with Event Counters and devices called Sequencers.

KOS makes Event Counters, Await Entries, and Message Queues available to EOS. It does not provide Locks, but it does provide Sequencers, so that EOS can construct its own Locks. The following discussion will define and explain the logical properties of Event Counters, Await Entries, Message Queues, Sequencers, and Locks. Their implementation in the present embodiment will be described along with the implementation of Processes 610 and Virtual Processors 612.

a. Event Counters 44801, Await Entries 44804, and Await Tables (Fig. 448, 449)

Event Counters, Await Entries, and Await Tables are the fundamental components of the KOS Synchronization System. Figure 448 illustrates Event Counters and Await Entries in the present embodiment. Figure 449 gives a simplified representation of Process Event Table 44705, the present embodiment's Await Tables. Turning first to Figure 448, Event Counter 44801 is an area of memory which

contains a value that may only be increased. In one of the present embodiment, Event Counters 44801 for KOS systems which may not page fault are always present in MEM 10112; other Event Counters 44801 are stored in Secondary Storage 10124 unless a Process 610 has referenced them and thereby caused the VMM System to load them into MEM 10112. The value contained in an Event Counter 44801 is termed an Event Counter Value 44802. In the present embodiment, EventCounter 44801 contains 64 bits of data, of which 60 make up Event Counter Value 44802. Event Counter 44801 may be referred to either as a variable or by means of a 128-bit UID pointer which contains Event Counter 44801's location. The UID pointer is termed an Event Counter Name 44803.

Await Entry 44804 is a component of entries in Await Tables. In the present embodiment, there are two Await Tables: Process Event Table 44705 and Virtual Processor Await Table (VPAT) 45401. VPAT 45401 is always present in MEM 10112. As already mentioned, Figure 449 illustrates PET 44705. Both PET 44705 and UPAT 45401 will be described in detail later. Each Await Entry 44804 contains an Event Counter Name 44803, an Event Counter Value 44802, and a Back Link 44805 which identifies a Process 610 or a Virtual Processor 612. Await Entry 44804 thus establishes a relationship between an Event Counter 44801, an Event Counter Value 44802, and a Process 610 or Virtual processor 612.

Turning now to Figure 449, in the present embodiment, all Await Entries 44804 for user Processes 610 are contained in PET 44705. PET 44705 also contains other information. Figure 449 presents only those parts of PET 44705 which illustrate Await Entries 44804. PET 44705 is structured to allow rapid location of Await Entries 44804 belonging to a specific Event Counter 44801. PET entries (PETEs) 44909 contain links which allow them to be combined into lists in PETE 44705. There are four kinds of lists in PET 44705:

- Event counter lists: these lists link all PETEs 44909 for Event Counters 44801 whose Event Counter Names 44803 hash to a single value.
- Await lists: These lists link all PETEs 44909 for Event Counters 44801 which a given Process 610 is awaiting.
- Interrupt lists: These lists link all PETEs 44909 for Event Counters 44801 which will cause an interrupt to occur for a given Process 610.
- The Free list: PETEs 44909 which are not being used in one of the above lists are on a free list.

Each PETE 44909 which is on an await list or an interrupt List is also on an event counter list.

Turning first to the event counter lists, all PETEs 44909 on a given event counter list contain Event Counter Names 44803 which hash to a single value. The value is produced by Hash Function 44901, and then used as an index in PET Hash Table (PETHT) 44903. That entry in PETHT 44903 contains the index in PET 44705 of that PETE 44909 which is the head of the event counter list. PETE List 44904 represents one such event counter list. Thus, given an Event Counter Name 44803, KOS can quickly find all Await Entries 44804 belonging to Event Counter 44801.

In the present embodiment, the implementation of Event Counters 44801 and tables with Await Entries 44804 involves both Processes 610 and Virtual Processors 612 to which Processes 610 are bound. As will be explained later, a large number of Event Counters 44801 and Await Entries 44804 belonging to Processes 610 are multiplexed onto a small number of Event Counters 44801 and Await Entries 44804 belonging to the Processes' Virtual Processors 612. Await entries 44804 for Event Counters 44801 belonging to Virtual Processors 612 are contained in VPAT 45401.

b. Synchronization with Event Counters 44801 and Await Entries 44804

The simplest form of Process 610 synchronization provided by KOS uses only Event Counters 44801 and Await Entries 44804. Coordination takes place like this: A Process 610 A requests KOS to perform an Await Operation, i.e., to establish one or more Await Entries 44804 and to suspend Process 610 A until one of the Await Entries is satisfied. In requesting the Await Operation, Process 610 A defines what Event Counters 44801 it is awaiting and what Event Counter Values 44802 these Event Counters 44801 must have for their Await Entries 44804 to be satisfied. After KOS establishes Await Entries 44804, it suspends Process 610 A. While process 610 A is suspended, other Processes 610 request KOS to perform Advance Operations on the Event Counters 44801 specified in Process 610 A's Await Entries 44804. Each time a Process 610 requests an Advance Operation on an Event Counter 44801, KOS increments Event Counter 44801 and checks Event Counter 44801's Await Entries 44804. Eventually, one Event Counter 44801 satisfies one of Process 610 A's Await Entries 44804, i.e., reaches a value equal to or greater than the Event Counter Value 44802 specified in its Await Entry 44804 for process 610 A. At this point, KOS allows process 610 A to resume execution. As process 610 A resumes execution, it deletes all of its Await Entries 44804.

E. Virtual Processors 612 (fig. 453)

As previously stated, a Virtual processor 612 may be logically defined as the means by which a Process 610 gains access to JP 10114. In physical terms, a Virtual Processor is an area of MEM 10112 which contains the information that the KOS microcode which binds Virtual Processors 612 to JP 10114 and unbinds them from JP 10114 requires to perform the binding and unbinding operations. Figure 453 shows a Virtual Processor 612. The area of MEM 10112 belonging to a Virtual Processor 612 is Virtual processor 612's Virtual Processor State Block (VPSB) 614. Each Virtual Processor 612 in a CS 10110 has a VPSB 614. Together, the VPSBs 614 make up VPSB Array 45301. Within the Virtual Processor management system, each Virtual Processor 612 is known by its VP Number 45304, which is the index of the Virtual Processor

612's VPSB 614 in VPSB Array 45301. Virtual Processors 612 are managed by means of lists contained in Micro VP Lists (MVPL) 45309. Each Virtual processor 612 has an Entry (MVPLE) 45321 in MVPL 45309, and as Virtual Processor 612 changes state, virtual processor management microcode moves it from one list to another in MVPL 45309.

5 VPSB 614 contains two kinds of information:

information from Process Object 901 belonging to Process 610 which is bound to VPSB 614's Virtual Processor 612, and information used by the Virtual Processor Management System to manage Virtual Processor 612. The most important information from Process Object 901 is the following:

- Process 610's principal and process UIDs 40401.
- 10 — AONs 41304 for Process 610's Stack Objects 44703. (VPSB 614 uses AONs 41304 because KOS guarantees that AONs 41304 belonging to Stack Objects 44703 will not change as long as a Process 610 is bound to a Virtual Processor 612.)

Given AON 41304 of Process 610's SS object 10336, the Virtual Processor Management System can locate that portion of Process 610's state which is moved into registers belonging to JP 10114 when process 15 610's Virtual Processor 612 is bound to JP 10114. Similarly, when Virtual Processor 612 is unbound from JP 10114, the virtual processor management system can move the contents of JP 10114 registers into the proper location in SS Object 10336.

a. Virtual Processor Management (Fig. 453)

20 EOS can perform six operations on Virtual Processors 612:

- Request VP allows EOS to request a Virtual Processor 612 from KOS.
- Release VP allows EOS to return a Virtual Processor 612 to KOS.
- Bind binds a Process 610 to a Virtual Processor 612.
- Unbind unbinds a process 610 from a Virtual Processor 612.
- 25 — Run allows KOS to bind Process 610's Virtual Processor 612 to JP 10114.
- Stop prevents KOS from binding process 610's Virtual Processor 612 to JP 10114.

As can be seen from the above list of operations, EOS has no direct influence over the actual binding of a Virtual Processor 612 to JP 10114. This operation is performed by a component of KOS microcode called the Dispatcher. Dispatcher microcode is executed whenever one of four things happens:

- 30 — Process 610 whose Virtual Processor 612 is currently bound to JP 10114 executes an Await Operation.
- Process 610 whose Virtual Processor 612 is currently bound to JP 10114 executes an Advance Operation which satisfies an Await Entry 44801 for some other Process 610.
- Either Interval Timer 25410 or Egg Timer 25412 overflows, causing an Event Signal which invokes Dispatcher microcode.
- 35 — IOJP Bus 10132 is activated, causing an Event Signal which invokes Dispatcher microcode. IOS 10116 activates IOJP bus 10132 when it loads data into MEM 10112 for JP 10114.

When Dispatcher microcode is invoked by one of these events, it examines lists in MVPL 45309 to determine which Virtual Processor 612 is to run next. For the purposes of the present discussion, only two lists are important: the running list and the eligible list. In the present embodiment, the running list, headed 40 by Running List Head 45321, contains only a single MVPLE 45321, that representing Virtual Processor 612 currently bound to JP 10114. In embodiments with multiple JPs 10114, the running list may have more than one MVPLE 45321. The eligible list, headed by Eligible List Head 45313, contains MVPLEs 45321 representing those Virtual Processors 612 which may be bound to JP 10114. MVPLEs 45321 on the eligible list are ordered by priorities assigned Processes 610 by EOS. Whenever KOS Dispatcher microcode is 45 invoked, it compares the priority of Process 610 whose Virtual Processor 612's MVPLE 45321 is on the running list with the priority of Process 610 whose Virtual Processor 612's MVPLE 45321 is at the head of the eligible list. If the latter Process 610 has a higher priority, KOS Dispatcher microcode places MVPLE 45321 belonging to the former Process 610's Virtual Processor 612 on the eligible list and MVPLE 45321 belonging to the latter Process 610's Virtual Processor 612 onto the running list. Dispatcher microcode then 50 swaps Processes 610 by moving state in JP 10114 belonging to the former Process 610 onto the former Process 610's SS object 10336 and moving JP 10114 state belonging to the latter Process 610 from the latter Process 610's SS object 10336 into JP 10114.

b. Virtual Processors 612 and Synchronization (Fig. 454)

55 When a synchronization operation is performed on a Process 610, one of the consequences of the operation is a synchronization operation on a Virtual Processor 612. For example, an Advance Operation which satisfies an Await Entry 44804 for a Process 610 causes an Advance Operation which satisfies a second Await Entry 44804 for Process 610's Virtual Processor 612. Similarly, a synchronization operation performed on a Virtual Processor 612 may have a synchronization operation on Virtual Processor 612's 60 Process 610 as a consequence. For example, if a Virtual Processor 612 performs an operation involving file I/O, Virtual Processor 612's Process 610 must await the completion of the I/O operation.

Figure 454 illustrates the means by which process level synchronization operations result in virtual processor-level synchronization operations and vice-versa. The discussion first describes the components which transmit process-level synchronization operations to Virtual Processors 612 and the manner in which 65 these components operate. Then it describes the components which transmit virtual processor-level

synchronization operations to Processes 610 and the operation of these components.

The first set of components is made up of VPSBA 45301 and VPAT 45401. VPSBA 45301 is shown here with two VPSBs 614: one belonging to a Virtual Processor 612 bound to a user Process 610 and one belonging to a Virtual Processor 612 bound to the KOS Process Manager process 610. VPAT 45401 is a virtual processor-level table of Await Entries 44804. Each Await Entry 44804 is contained in a VPAT Entry (VPATE) 45403. Each Virtual Processor 612 bound to a Process 610 has a VPAT Chunk 45402 of four VPATEs 45403 in VPAT 45401, and can thus await up to four Event Counters 44801 at any given time. The location of a Virtual processor 612's VPAT Chunk 45402 is kept in Virtual Processor 612's VPSB 614. When an Advance Operation satisfies any of the Await Entries 44804 belonging to a Virtual Processor 612, all in Virtual Processor 612's VAT Chunk 45402's Await Entries 44804 are deleted. As in PET 44705, VPATEs 45403 containing Await Entries 44804 which are awaiting a given Event Counter 44801 are linked together in a list.

VPATEs 45403 for Virtual Processors 612 bound to user Processes 610 may contain Await Entries 44804 for user Process 610's Private Event Counter 45405. Private Event Counter 45405 is contained in Process 610's Process Object 901. It is advanced each time an Await Entry 44804 in a PETE 44909 on a PET List belonging to Process 610 is satisfied.

The components operate as follows: When KOS performs an Await Operation on Process 610, it makes Await Entries 44804 in both PET 44705 and VPAT 45401 and puts Process 610's VP 612 on the suspended list in MVPL 45309. As previously described, an Await Entry 44804 in PET 44705 awaits an Event Counter 44801 specified in the Await Operation which created Await Entry 44804. Await Entry 44804 in VPAT 45401 awaits Process 610's Private Event Counter 45405. Each time an Await Entry 44804 belonging to Process 610 in PET 44705 is satisfied, Process 610's Private Event Counter 45405 is advanced. The advance of Private Event Counter 45405 satisfies Await Entry 44801 for Process 610's Virtual processor 612 in VPAT 45401, and consequently, KOS deletes Virtual Processor 612's VPATEs 45403 and moves Virtual Processor 612's MVPLE 45321 in MVPL 45309 from the suspended list to the eligible list.

The components which allow a Virtual Processor 612 to transmit a synchronization operation to a process 610 are the following: Outward Signals Object (OSO) 45409, Multiplexed Outward Signals Event Counter 45407, and PET 44705. OSO 45409 contains Event Counters 44801 which KOS FU 10120 microcode advances when it performs operations which user Processes 610 are awaiting. Event Counters 44801 in OSO 45409 are awaited by Await Entries 44804 in PET 44705. Each time KOS FU 10120 microcode advances an Event Counter 44801 in OSO 45409, it also advances Multiplexed Outward Signals Event Counter 45407. It is awaited by an Await Entry 44804 in VPAT 45401 belonging to Virtual Processor 612 bound to KOS Process Manager Process 610. When Virtual Processor 612 bound to KOS Process Manager Process 610 is again bound to JP 10114, KOS Process Manager Process 610 examines all PETEs 44909 belonging to the Event Counters 44801 in OSO 45423. If an advance of an Event Counter 44801 in OSO 44801 satisfied a PETE 44909 Process 610, that Process 610's Private Event Counter 45405 is advanced as previously described, and Process 610 may again execute.

A user I/O operation illustrates how the components work together. Each user I/O channel has an Event Counter 44801 in OSO 45409. When a Process 610 performs a user I/O operation on a channel, the EOS I/O routine establish an Await Entry 44804 in the PET 44705 list belonging to Process 610 for the channel's Event Counter 44801 in OSO 45409. When the I/O operation is complete, IOS 10116 places a message to JP 10114 in an area of MEM 10112 and activates IOJP Bus 10132. The activation of IOJP Bus 10132 causes an Event Signal which invokes KOS microcode. The microcode examines the message from IOS 10116 to determine which channel is involved, and then advances Event Counter 44801 for that channel in OSO 45409 and Multiplexed Outward Signals Event Counter 45407. The latter advance satisfies an Await Entry 44804 for Process Manager Process 610's Virtual Processor 612 in VPAT 45401, and Process Manager Process 610 begins executing. Process Manager Process 610 examines OSO 45409 to determine which Event Counters 44801 in OSO 45409 have been advanced since the last time process manager Process 610 executed, and when it finds such an Event Counter 44801, it examines the Event Counter Chain in PET 44705 for that Event Counter 44801. If it finds that the advance satisfied any Await Entries 44804 in the Event Counter Chain, it advances Private Event Counter 45405 belonging to Process 610 specified in Await Entry 44804, thereby causing that Process 610 to resume execution as previously described.

F. Process 610 Stack Manipulation

This section of the specification for CS 10110 describes the manner in which Process 610's MAS 502 and SS 504 are manipulated. As previously mentioned, in CS 10110, a Process 610's MAS 502 and SS 504 are contained in several objects. In the present embodiment, there are five objects, one for each domain's portion of the Macro Stack (MAS) (MAS Objects 10328 through 10324) and one for the Secure Stack (SS) (SS Object 10336). In other embodiments, a Process 610's MAS 502 and SS 504 may contain objects for user-defined domains as well. Though a Process 610's MAS 502 and SS 504 are contained in many objects, they function as a single logical stack. The division into several objects is a consequence of two things: the domain component of the protection system, which requires that an object referenced by a Procedure 602 have Procedure 602's domain of execution, and the need for a location inaccessible to user programs for micromachine state and state which may be manipulated only by KOS.

Stack manipulation takes place under the following circumstances:

- When a procedure 602 is invoked or a Return SIN is executed. Procedure 602 invocations are

performed by means of a Call SIN. Call causes a transfer of control to the first SIN in the invoked Procedure 602 and the Return SIN causes a transfer of control back to the SIN in the invoking Procedure 602 which follows the Call SIN.

- 5 — When a non-local Go To SIN is executed. The non-local Go To causes a transfer of control to an arbitrary position in some Procedure 602 which was previously invoked by Process 610 and whose invocation has not yet ended.
 - When a condition arises, i.e., an execution of a statement in a program puts the executive Process 610 into a state which requires the execution of a previously established Handler Procedure 602.
 - 10 — When a Process 610 is interrupted, i.e., when an Interrupt Entry 45718 for Process 610 is satisfied.
- Most of the mechanisms involved in stack manipulation are used in Call and Return; these operations are therefore dealt with in detail and the other operations only as they differ from Call and Return. The discussion first introduces Call and Return, then explains the stacks in detail, and finally analyzes Call and Return and the other operations in detail.

15 1. Introduction to Call and Return

As a Process 610 executes a program, it executes Call and Return SINs. A Call SIN begins an invocation of a procedure 602, and a Return SIN ends the invocation. Generally speaking, a Call SIN does the following:

- 20 — It saves the state of Process 610's execution of Procedure 602 which contains the Call SIN. Included in this state is the information required to continue Procedure 602's execution after the Call SIN is finished. This portion of the state is termed calling Procedure 602's Macrostate.
- It creates the state which Process 610 requires to begin execution called Procedure 602.
- It transfers control to the first SIN in the called Procedure 602's code.

The Return SIN does the opposite: it releases the state of called Procedure 602, restores the saved state of calling Procedure 602, and transfers control to the SIN in the calling Procedure 602 following the Call SIN. An invocation of a Procedure 602 lasts from the execution of the Call SIN which transfers control to the Procedure 602 to the execution of the Return SIN which transfers control back to Procedure 602 which contained the Call SIN. The state belonging to a given invocation of a Procedure 602 by a Process 610 is called Procedure 602's invocation state.

30 While Calls and Returns may be implemented in many different fashions, it is advantageous to implement them using stacks. When a Call creates invocation state for a Procedure 602, that invocation state is added to the top of Process 610's stack. The area of a stack which contains the invocation state of a Procedure 602 is called a frame. Since a called Procedure 602 may call another procedure 602, and that another, a stack may have any number of frames, each frame containing the invocation state resulting from the invocation of a Procedure 602 by Process 610, and each frame lasting as long as the invocation it represents. When called Procedure 602 returns to its caller, the frame upon which it executes is released and the caller resumes execution on its frame. Procedure 602 being currently executed by a Process 610 thus always runs on the top frame of Process 610's MAS 502.

40 Calls and Returns in CS 10110 behave logically like those in other computer systems using stacks to preserve process 610 state. When a Process 610 executes a Call SIN, the SIN saves as Macrostate the current values of the ABPs, the location of the SIN at which the execution of calling Procedure 602 is to continue, and information such as a pointer to calling Procedure 602's Name Table 10350 and UID 40401 belonging to the S-interpreter object which contains the S-interpreter for Procedure 602's S-language. The Call SIN then creates a stack frame for called Procedure 602, obtains the proper ABP values, the location of called Procedure 602's Name Table 10350 and UID 40401 belonging to its S-interpreter object, and begins executing newly-invoked Procedure 602 on the newly-created stack frame. The Return SIN deletes the stack frame obtains the ABP values and name interpreter information from the Macrostate saved during the Call SIN and then transfers control to the SIN at which execution of calling Procedure 602 is to continue.

50 However the manner in which Call and Return are implemented is deeply affected by CS 10110's Access Control System. Broadly speaking there are two classes of Calls and Returns in CS 0110: those which are mediated by KOS and those which are not. In the following discussion, the former class of Calls and Returns are termed Mediated Calls and Returns, and the latter are called Neighborhood Calls and Returns. Most Calls and Returns executed by CS 10110 are Neighborhood Calls and Returns; Mediated Calls and Returns are typically executed when a user procedure 602 calls EOS Procedures 602 and these in turn call KOS Procedures 602. The Mediated Call makes CS 10110 facilities available to user Processes 610 while protecting these CS 10110 facilities from misuse and therefore generally serves the same purpose as system calls in the present art. As will be seen in the ensuing discussion, Mediated Call requires more CS 10110 overhead than Neighborhood Call but the extra overhead is less than that generally required by system calls in the present art.

60 Mediated Calls and Returns involve S-interpreter, Namespace, and KOS microcode. S-interpreter and Namespace microcode interpret the Names involved in the call and only modifies those portions of Macrostate accessible to the S-interpreter. The remaining Macrostate is modified by KOS microroutines invoked in the course of the Call SIN. A Mediated Call may be made to any Procedure 602 contained in an object to which Process 610's subject has Execute Access at the time the invocation occurs. Mediated Calls and Returns must be made in the following situations:

- When called Procedure 602 has a different Procedure Environment Descriptor (PED) 30303 from that used by calling Procedure 602. Such Calls are termed Cross-PED Calls.
- When called Procedure 602 is in a different Procedure Object 608 from calling Procedure 602. Such Calls are termed Cross-Procedure Object Calls.

- 8 — When called Procedure 602's Procedure Object 608 has a different Domain of Execution (DOE) Attribute from that of calling Procedure 602's Procedure Object 608, and therefore must place its Invocation State on a different MAS object from that used by calling Procedure 602. Such Calls are termed Cross-Domain Calls.

10 In all of the above Calls, the information required to complete the Call is not available to the S-interpreter and consequently, KOS mediation is required to complete the Call. Neighborhood Calls and Returns only modify two components of Macrostate: the pointer to the current SIN and the FP ABP. Both of these components are available to the S-interpreter as long as called Procedure 602 has the same PED 30303 i.e., uses the same Name Tab 10350 and S-interpreter or the calling Procedure 602 and has Names with the same syllable size as calling Procedure 602. The Call and Return SINS are specific to each S-language, but they resemble each other in their general behavior. The following discussion will deal exclusively with this general behavior and will concentrate on Mediated Calls and Returns. The discussion first describes MAS 502 and SS 504 belonging to a Process 610 and those parts of Procedure Object 608 involved in Calls and Returns, and then describes the implementation of Calls and Returns.

20 **2. Macro Stacks (MAS) 502 (Fig. 467)**

Figure 467 gives an overview of an object belonging to a Process 610's MAS 502. The description of this Figure will be followed by descriptions of other Figures containing detailed representations of portions of MAS objects.

25 At a minimum MAS Object 46703 comprises KOS MAS Header 10410 together with Unused Storage 46727 reserved for the other elements comprising MAS Object 46703. If Process 610 has not yet returned from an invocation of a Procedure 602 contained in a Procedure Object 608 whose DOE is that required for access to MAS Object 46703. MAS object 46703 further comprises a Stack Base 46703 and at least one MAS Frame 46709.

30 Each MAS Frame 46709 represents one mediated invocation of a procedure 602 contained in a Procedure Object 608 with the DOE attribute required by MAS 46703, and may in addition represent neighborhood invocations of Procedures 602 which share that Procedure 602's Procedure Object 608. The topmost MAS Frame 46709 represents the most recent group of invocations of Procedures 602 with the DOE attribute required by MAS Object 46703 and the bottom MAS Frame 46709 the earliest group of invocations from which Process 610 has not yet returned. Frames for invocations of Procedures 602 with other domains of execution are contained in other MAS Objects 46703. As will be explained in detail below MAS Frames 46709 in different MAS objects 46703 are linked by pointers.

35 MAS Domain Stack Base 46703 has two main parts: KOS MAS Header 10410 which contains information used by KOS microcode which manipulates MAS Object 46703, and Perdomain Information 46707, which contains information about 46703's domain and static information, i.e., information which lasts longer than an invocation used by Procedures 602 with MAS Frames 46709 on MAS Object 46703. MAS Frame 46709 also has two main parts, a KOS Frame Header 10414 which contains information used by KOS to manipulate Frame 46709 and S-interpreter Portion 46713 which contains information available to the S-interpreter when it executes the group of Procedures 602 whose invocations are represented by Frame 46709.

45 When making Calls and Returns, the S-interpreter and KOS microcode use a group of pointers to locations in MAS Object 46703. These pointers comprise the following:

- MAS Object UID 46715 the UID 40401 of AS Object 46703.
- First Frame Offset (FFO) 46719 which locates the beginning of KOS Frame Header 10414 belonging to the first MAS Frame 46709 in MAS Object 46703.
- 50 — Frame Header Pointer (FHP) 46702 which locates the beginning of the topmost KOS Frame Header 10414 in MAS Object 46703.
- Stack Top Offset (STO) 46704 a 32-bit offset from Stack UID 46715 which marks the first bit in Unused Storage 46727.

As will be seen presently all of these pointers are contained in fields in KOS MAS Header 46705.

55 **a.a. MAS Base 10410 (Fig. 468)**

Figure 468 is a detailed representation of MAS Domain Stack Base 10410 Turning first to the detailed representation of KOS MAS Header 46705 contained therein, there are the following fields:

- Format Information Field 46801 containing information about the format of KOS MAS Header 46705.
- 60 — Flags Field 46803. Of these flags, only one is of interest to the present discussion: Domain Active Flag 46804. This flag is set to TRUE when Process 610 to which MAS Object 46703 belongs is executing the invocation of Procedure 602 whose invocation record makes up the topmost MAS Frame 46709 contained in MAS Object 46703 to which KOS MAS Header 46705 belongs.
- PFO Field 46805: All MAS Headers 46705 and Frame Headers 46709 have fields containing offsets locating the previous and following headers in MAS Object 46703. In a Stack Header 46705 there is no

65

previous header and this field is set to 0.

- FFO Field 46805: The field locating the following header in a Stack Header 46705 this field contains FFO 46719 since the next header is the first Frame Header in MAS Object 46703.
 - STO Field 46807: the field containing STO offset 46704.
 - 5 — Process ID Field 46809: UID 40401 belonging to Process Object 901 for Process 610 to which MAS Object 46703 belongs.
 - Domain Environment Information pointer Field 46811: The pointer contained in the field locates an area which contains domain-specific information. In the present embodiment, the area is part of MAS Stack Base 10410; however, in other embodiments, it may be contained in a separate object.
 - 10 — Signaller Pointer Field 46813: The pointer contained in the field locates a Procedure 602 which KOS invokes when a Process 610's execution causes a condition to arise while it is executing in the domain to which MAS object 46703 belongs.
 - AAT Pointer Field 30211: The pointer in Field 30211 locates AAT 30201 for MAS Object 46703. AAT 30201 is described in detail in Chapter 3.
 - 15 — Frame Label Sequencer Field 46819: This field contains a Sequencer 45102. Sequencer 45102 is used to generate labels used to locate MAS Frames 46709 when a non-local GOTO is executed.
- Turning now to the detailed representation of Domain Environment Information 46821 located by Domain Environment Information Pointer Field 46811 there are the following fields:
- KOS Format Information Field 46823.
 - 20 — Flags Field 46825 containing the following flags:
 - Pending Interrupt Flag 46827, set to TRUE when Process 610 has an interrupt pending for the domain to which MAS Object 46703 belongs.
 - Domain Dead Flag 46829, set to TRUE when Process 610 can no longer execute Procedures 602 with domains of execution equal to that to which MAS Object 46703 belongs.
 - 25 — Invoke Verify on Entry Flag 46833 and Invoke Verify on Exit Flag 46835. The former flag is set to TRUE when KOS is to invoke a Procedure 602 which checks the domain's data bases before a Procedure 602 is allowed to execute on the domain's MAS Object 46703; the latter is set to TRUE when KOS is to invoke such a Procedure 602 on exit from a Procedure 602 with the domain as its DOE.
 - 30 — Default Handler Non-null Flag 46835 is set to TRUE when there is a default clean-up handler for the domain. Clean-up handlers are described later.
 - Interrupt Mask Field 46839 determines what interrupts set for Process 610 in MAS object 46703's domain will be honored.
 - Domain UID Field 46841 contains UID 40401 for the domain to which MAS Object 46703 belongs.
 - 35 — Fields 46843 through 46849 are pointers to Procedures 602 or tables of pointers to Procedures 602. The Procedures 602 so located handle situations which arise as MASs 502 are manipulated. The use of these fields will become clear as the operations which require their use are explained.

b.b. Per-domain-Data Area 46853 (Fig. 468)

40 Per-domain Data Area 46853 contains data which cannot be kept in MAS Frames 46709 belonging to invocations of Procedures 602 executing in MAS Object 46703's domain, but which must be available to these invocations Per-Domain Data Area 46853 has two components: Storage Area 46854 and AAT 30201. Storage Area 46854 contains static data used by Procedures 602 with invocations on MAS Object 46703 and data used by S-interpreters which are used by such procedures 602. Associated Address Table (AAT) 30201 is used to locate data in Storage Area 46854. A detailed discussion of AAT 30201 is contained in Chapter 3.

Two kinds of data is stored in Storage Area 46854: static data and S-interpreter data.

Static data is stored in Static Data Block 46863. Static Data Block 46863 comprises two parts: Linkage Pointers 46865 and Static Data Storage 46867 Linkage Pointers 46865 are pointers to static data not contained in Static Data Storage 46867 for example, data which lasts longer than Process 610 and pointers to External Procedures 602 which the Procedure 602 to which Static Data Storage 46867 belongs invokes. 50 Static Data Storage 46867 contains storage for static data used by the Procedure 602 which does not last longer than Process 610 executing the Procedure 602.

S-interpreter data is data required by S-interpreters used by Procedures 602 executing on MAS object 46703.

55 The S-interpreter data is stored in S-interpreter Environment Block (SEB) 46864 which, like Static Data Block 46864 is located via AAT 30201: The contents of SEB 46864 depend on the S-interpreter.

c.c. MAS Frame 46709 Detail (fig. 469)

Figure 469 represents a typical frame in MAS Object 46703. Each MAS Frame 46709 contains a 60 Mediated Frame 46947 produced by a Mediated Call of a Procedure 602 contained in a Procedure Object 608 whose DOE attribute is the one required for execution on MAS object 46703. Mediated Frame 46947 may be followed by Neighborhood Frames 46945 produced by Neighborhood Calls of Procedures 602. Mediated Frame 46947 has two parts, a KOS Frame Header 10414 which is manipulated by KOS microcode, and an S-interpreter portion which is manipulated by S-interpreter and Namespace microcode. 65 Neighborhood Frames 46945 have no KOS Frame Headers 10414. As will become clear upon closer

EP 0 067 556 B1

5 examination of Figure 469. Mediated Frames 46947 in the present embodiment contain no Macrostate. In the present embodiment, Macrostate for these frames is kept on SS Object 10336; however in other embodiments, Macrostate may be stored in Mediated Frames 46947. Neighborhood Frames 46945 contain those portions of the macrostate which may be manipulated by Neighborhood Call; the location of this macrostate depends on the Neighborhood Call SIN.

Turning now to KOS Frame Header 10414, there are the following fields:

- KOS Format Information Field 46901 containing information about MAS Frame 46709's format.
- Flags Field 46902. This field contains the following flags:
 - Result of Cross-domain Call Flag 46903. This flag is TRUE if MAS Frame 46709 which precedes this MAS Frame 46709 is in another MAS Object 46703.
 - Is Signaller Flag 46905. This flag is TRUE if this MAS Frame 46709 was created by the invocation of a Signaller Procedure 602.
 - Do Not Return Flag 46907: This flag is TRUE if Process 610 is not to return to the invocation for which this MAS Frame 46709 was created.
 - Flags 46909 through 46915 indicate whether various lists used in condition handling and non-local GOTOs are present in the MAS Frame 46709.
 - Previous Frame Offset Field 46917. Next Frame Offset Field 46919. and Frame Top Offset Field 46921 are offsets which give the location where Header 10414 for the previous MAS Frame 46709 in MAS Object 46703 begins, the location where the header for the next MAS Frame 46709 in MAS Object 46703 begins, and the location of the first bit beyond the top of MAS Frame 46709 respectively.
 - Fields 46923 through 46927 are offsets which locate lists in S-interpreter portion 46713 of Frame 46709. KOS establishes such lists to handle conditions and non-local GOTOs. Their use will be explained in detail under those headings.
 - Fields 46929 and 46933 contain information about Procedure 602 whose invocation is represented by MAS Frame 46709. Field 46929 contains the number of arguments required by procedure 602 and Field 46933 contains a resolvable pointer to Procedure 602's PED 30303. Both these fields are used primarily for debugging.
 - Dynamic Back Pointer Field 46931 contains a resolvable pointer to the preceding MAS Frame 46709 belonging to Process 610's MAS 502 when that MAS Frame 46709 is contained in a different MAS Object 46703. In this case, Flag Field 46903 is set to TRUE. When the preceding MAS Frame 46709 is contained in the same MAS object 46703 field 46931 contains a pointer with a null UID 40401 and Flag Field 46903 is set to FALSE.
 - Frame Label Field 46935 is for a Frame Label produced when a non-local GOTO is established which transfers control to the invocation represented by MAS Frame 46709. The label is generated by Frame Label Sequencer 46819 in KOS MAS Header 10410.
- S-interpreter Portion 46713 of MAS Frame 46709 comprises those portions of MAS Frame 46709 which are under control of the S-interpreter. S-interpreter Portion 46713 in turn comprises two main subdivisions: those parts belonging to Mediated Frame 46947 and those belonging to Neighborhood Frames 46945.
- The exact form of S-interpreter portion 46949 of KOS Frame 46947 and of S-interpreter Frames 46945 depends on the Call SIN which created the frame in question. However all Neighborhood Frames 46945 and S-interpreter portions 46949 of Mediated Frames 46947 have the same arrangements for storing Linkage Pointers 10416 and local data in the frame. Linkage Pointers 10416 are pointers to the locations of actual arguments used in the invocation and Local Storage 10420 contains data which exists only during the invocation. In all Mediated Frames 46947 and Neighborhood Frames 46945. Linkage pointers 10416 precede Local Storage 10420. Furthermore, when a Mediated Frame 46947 or a Neighborhood Frame 46945 is the topmost frame of Process 610's MAS, i.e. when Process 610 is executing on that frame, the FP always points to the beginning of Local Storage 10420, and the beginning of Linkage Pointers 10416 is always at a known displacement from FP. References to Linkage Pointers 10416 may therefore be expressed as negative offsets from FP, and references to Local Storage 10420 as positive offsets.
- In addition, S-interpreter Portion 46713 may contain lists of information used by KOS to execute non-local GOTOs and conditions, as well as S-interpreter frames for non-mediated calls. The lists of information used by KOS are contained in List Area 46943. The exact location of List Area 46943 is determined by the compiler which generates the SINs and Name Table for the Procedure 602 whose invocation is represented by Mediated Frame 46947. When Procedure 602's source text contains statements requiring storage in List Area 46943, the compiler generates SINs which place the required amount of storage in Local Storage 10420. KOS routines then build lists in Area 46943, and place the offsets of the heads of the lists in Fields 46923, 46925 or 46927, depending on the kind of list. The lists and their uses are described in detail later.

3. SS 504 (Fig. 470)

60 Figure 470 presents an overview of SS 504 belonging to a Process 610. SS 504 is contained in SS Object 10336. SS Object 10336 is manipulated only by KOS microcode routines. Neither Procedures 602 being executed by Process 610 nor S-interpreter or Namespace microcode may access information contained in SS Object 10336.

65 SS Object 10336 comprises two main components, SS Base 47001 and SS Frames 47003. Turning first to the general structure of SS Frames 47003, each time a Process 610 executes a Mediated Call KOS

microcode creates a new SS Frame 47003 on SS Object 10336 belonging to Process 610 and each time a Process 610 executes a Mediated Return, KOS microcode removes the current top SS Frame 47003 from SS Object 10336. There is thus one SS Frame 47003 on SS Object 10336 belonging to a process 610 for each Mediated Frame 46947 on Process 610's MAS 502.

5 SS Frames 47003 comprise two kinds of frames:

Ordinary Frames 10510 and Cross-domain Frames 47039. Cross-domain Frames 47039 are created whenever Process 610 executes a Cross-domain Call; for all other Mediated Calls. Ordinary Frames 10510 are created. Cross-domain Frames 47039 divide SS Frames 47003 into Groups 47037 of SS Frames 47003 belonging to sequences of invocations in a single domain. The first SS Frame 47003 in a Group 47037 is a Cross-domain Frame 47039 for the invocation which entered the domain, and the remainder of the SS Frames 47003 are Ordinary Frames 10510 for a sequence of invocations in that domain. These groups of SS Frames 47003 correspond to groups of Mediated Frames 46947 in a single MAS Object 46703.

a.a. SS Base 47001 (Fig. 471)

15 SS Base 47001 comprises four main parts: SS Header 10512 Process Microstate 47017, Storage Area 47033 for JP 10114 register contents, and Initialization Frame Header 47035. Secure Stack Header 10512 contains the following information:

- Fields 47001 and 47009 contain flag and format information; the exact contents of these fields are unimportant to the present discussion.
- 20 — Previous Frame Offset Value Field 47011 is a standard field in headers in SS Object 10336; here it is set to 0, since there is no previous frame.
- Secure Stack First Frame Offset Field 47013 contains the offset of the first SS Frame 47039 in SS object 10336, i.e., Initialization Frame Header 47035.
- Process UID field 47015 contains UID 40401 of Process 610 to which SS Object 10336 belongs.
- 25 — Number of Cross Domain Frames Field 47016 contains the number of Cross-domain Frames 47039 in SS Object 10336.

Process Microstate 47017 contains information used by KOS microcode when it executes Process 610 to which SS Object 10336 belongs. Fields 47019, 47021 and 47022 contain the offsets of locations in SS Object 10336. Field 47019 contains the value of SSTO the location of the first free bit in SS Object 10336; 30 Field 47021 contains the value of SSFO, the location of the topmost frame in SS object 10336; Field 47022 finally contains the value of XDFO, the location of the topmost Cross-domain Frame 47039 in SS Object 10336. All of these locations are marked in Figure 470.

Other fields of interest in Process Microstate 47017 comprise the following: Offsets in Storage Area Field 47023 contains offsets of locations in Storage Area 47033 of SS Object 10336; Domain Number Field 35 47025 contains the domain number for the DOE of Procedure 602 currently being executed by Process 610. The relationship between domain UIDs and domain numbers is explained in the discussion of domains. VPAT Offset Field 47027 contains the offset in VPAT 45401 of VPAT Chunk 45402 belonging to Virtual Processor 612 to which Process 610 is bound. Signal Pointer Field 47029 contains a resolved pointer to the Signaller (a Procedure 602 used in condition handling) belonging to the domain specified by Domain 40 Number Field 47025 and Trace Information Field 47031 contains a resolved pointer to that domain's Trace Table, described later.

Storage Area for JP 10114 register Contents 47033 is used when a Virtual Processor 612 must be removed from JP 10114. When this occurs, either because Virtual Processor 612 is unbound from JP 10114, 45 because CS 10110 is being halted, or because CS 10110 has failed, the contents of JP 10114 registers which contain information specific to Virtual Processor 612 are copied into Storage Area 47033. When Virtual Processor 612 is returned to JP 10114, these register contents are loaded back into the JP 10114 registers from whence they came. Initialization Frame Header 47035, finally, is a dummy frame header which is used in the creation of SS Object 10336.

50 b.b. SS Frames 47003 (Fig. 471)

Commencing the discussion of SS Frames 47039 and 10510. Figure 471 illustrates these structures in detail. Ordinary SS Frame 10510 comprises three main divisions: Ordinary SS Frame Header 10514, Macrostate 10516 and Microstate 10520. Ordinary SS Frame Header 10514 contains information used by KOS microcode to manipulate Ordinary SS Frame 10510 to which Header 10514 belongs. Macrostate 10516 55 contains the values of the ABPs for the frame's mediated invocation and other information required to resume execution of the invocation. Microstate 10520 contains micromachine state from FU 10120 and EU 10122 registers. The amount of micromachine state depends on the circumstances; in the present embodiment, some micromachine state is saved on all Mediated Calls; furthermore, if a Process 610 executes a microcode-to-software Call, the micromachine state that existed at the time of the call is saved; 60 finally, Microstate 10520 belonging to the topmost SS Frame 47003 may contain information which was transferred from FU 10120 GRF registers 10354 or EU 10122 register and stack mechanism 10216 when their capacity was exceeded. For details about this portion of Microstate 10520 see the discussion of the FU 10120 micromachine in Chapter 2. The discussion of SS Object 10336 continues with details concerning SS Header 10514 and Macrostate O5163.

65

a.a.a. Ordinary SS Frame Headers 10514 (Fig. 741)

Fields of interest in Ordinary Secure Stack Frame Header 10514 are the following:

- Format Information 47103 which identifies the format of Header 10514.
- Flags Field 47105 which contains one flag of interest in this discussion: Frame Type Flag 47107: in Ordinary SS Frames 10510 this field is set to FALSE.
- Offset Fields 47109 through 47113: Field 47109 contains the offset of the previous SS Frame 47039 or 10510. Field 47111 contains the offset of the following SS Frame 47039 or 10510. and Field 47113 contains the offset of the last SS Frame 47039 or 10510 preceding the next Crossdomain Frame 47039
- Field 47117 contains the current domain number for the domain in which the mediated invocation represent SS Frame 47039 or 10510 is executing.
- Field 47119 contains the offset of the preceding Cross-domain Frame 47039.
- Field 47121 contains offsets for important locations in Microstate 10520.

b.b.b. Detailed Structure of Macrostate 10516 (Fig. 471)

These fields are of interest in Macrostate 10516:

- Syllable Size Field 47125 contains the value of K, i.e., the size of the Names in the SINs belonging to Procedure 602 which the invocation is executing.
- End of Name Table Field 47127 contains the location of the last Name in Name Table 10350 belonging to Procedure 602 which the invocation is executing.
- Fields 47129 through 47143 are resolved pointers to locations in Procedure Object 901 containing Procedure 602 being executed by the invocation and resolved pointers to locations containing data being used by Procedure 602. Field 47129 contains a pointer to Procedure 602's PED 30303; if Procedure 602 is an External Procedure 602. Field 47131 contains a pointer to Procedure 602's entry in Gates 10340; Field 47135 contains the UID-offset value of FP for the invocation; Field 47135 contains a pointer to SEB 46864 used by Procedure 602's S-interpreter. Field 47137 contains the UID-offset value of SDP and Field 47139 contains that of PBP. SIP Field 47141 contains a pointer to Procedure 602's S-interpreter object, and NTP, finally, is a pointer to Procedure 602's Name Table 10350.
- Field 47145 contains the PC for the SIN which is to be executed on return from the mediated invocation to which SS Frame 47003 belongs.

c.c.c. Cross domain SS Frames 47039 (Fig. 471)

Cross-domain SS Frames 47039 differ from Ordinary SS Frames 10510 in two respects: they have an additional component, Cross-domain State 10513, and fields in Cross domain Frame Header 47157 have different meanings from those in Ordinary Frame Header 10514.

Cross-domain State 10513 contains information which KOS Call microcode uses to verify that a return to a Procedure 602 whose DOE differs from that of Procedure 602 whose invocation has ended is returning to the proper domain. Fields of interest in Cross-domain State 10513 include GOTO Tag 47155 used for non-local GOTOs which cross domains, Stack Top Pointer Value 47153, which gives the location of the first free bit in the new domain's MAS Object 46703 and Frame Header Pointer Value 47151, which contains the location of the topmost Mediated Frame Header 46709 in new MAS Object 46703.

There are three fields in Cross-domain Frame Header 47157 which differ from those in Ordinary SS Frame Header 47101. These fields are Flag Field 47107 which in Cross-domain Frame Header 47157 always has the value TRUE, preceding Cross-domain Frame Offset Field 47161, which contains the offset of preceding Cross-domain Frame 47039 in SS Object 10336 and Next Cross domain Frame Offset Field 47159, which contains the location of the next Cross-domain Frame 47039. These last two fields occupy the same locations as Fields 47111 and 47109 respectively in Ordinary SS Frame Header 10514.

As will be noted from the above description of SS Frames 47003. Secure Stack Object 10336 in the present embodiment contains three kinds of information: macrostate cross-domain state and microstate. In other embodiments, the information in SS object 10336 may be stored in separate stack structures, for example, separate microstate and cross-domain stacks, or information presently stored in MAS Objects 46703 may be stored in SS Object 10336, and vice-versa.

4. Portion of Procedure Object 608 Relevant to Call and Return (Fig. 472)

The information which Process 610 requires to construct new frames on its MAS Objects 46703 and SS Object 10336 and to transfer control to invoked Procedure 602 is contained in invoked Procedure 602's Procedure Object 608. Figure 472 is an overview of Procedure Object 608 showing the information used in a Call. Figure 472 expands information contained in Figures 103 and 303; fields that appear in those Figures have the names and numbers used there.

Beginning with Procedure Object Header 10336, this area contains two items of information used in Calls: an offset in Field 47201 giving the location of Argument Information Array 10352 in Procedure Object 608 and a value in Field 47203 specifying the number of gates in Procedure Object 608. Gates allow the invocation of External Procedures 602 that is, Procedures 602 which may be invoked by Procedures 602 contained in other Procedure Objects 608. Procedure Object 608's gates are contained in External Entry Descriptor Area 10340. There are two kinds of gates: those for Procedures 602 contained in Procedure Object 608, and those for procedures 602 contained in other Procedure Objects 608, but callable via

Procedure Object 608. Gates for Procedures 602 contained in Procedure Object 608 are termed Local Gates 47205. Local Gates 47205 contain Internal Entry Offset (IEO) Field 47207 which contains the offset in Procedure Object 608 of Entry Descriptor 47227 for Procedure 602. If Procedure 602 is not contained in Procedure Object 472 its gate is a Link Gate 47206. Link Gates 47206 contain Binder Area Pointer (BAP) Fields 47208. A BAP Field 47208 contains the locations of an area in Binder Area 30323 which in turn contains a pointer to a Gate in another Procedure Object 608. The pointer in Binder Area 30323 may be either resolved or unresolved. If Procedure 602 is contained in that Procedure Object 608, the Gate is a Local Gate 47205; otherwise, it is another Link Gate 47206.

Procedure Environment Descriptors (PEDS) 10348 contains PEDs 30303 for Procedures 602 contained in Procedure Object 608. Most of the macrostate information for a Procedure 602 may be found in its PED 30303. PED 30303 has already been described, but for ease of understanding, its contents are reviewed here.

- K Field 30305 contains the size of Procedure 602's Names.
- Largest Name (LN) Field 30307 contains the i

Beginning with Procedure Object Header 10336, this area contains two items of information used in Calls: an Offset in Field 47201 giving the location of Argument Information Array 10352 in Procedure Object 608 and a value in Field 47203 specifying the number of gates in Procedure Object 608. Gates allow the invocation of External Procedures 602, that is, Procedures 602 which may be invoked by Procedures 602 contained in other Procedure Objects enter to Static Data Block 46863. Thus, for that invocation of Procedure 602 on invocation, the SDP ABP is derived via SDPP field 30313.

- PBP Field 30315 is the pointer from which the current PC is calculated. When Procedure 602 is invoked, this value becomes the PBP ABP.
- S-Interpreter Environment Prototype Pointer (SEPP) Field 30316 contains the location of SEB Prototype Field 30317. When Procedure 602 is invoked, Field 30316 locates SEB 46864 via AAT 30201 in the same manner as SDPP field 30313 locates the invocation's static data.

A Procedure 602's PED 30303 may be located from its Internal Entry Descriptor 47227. A PED 30303 may be shared by several Procedures 602. Of course in this case, the values contained in shared PED 30303 are the same for all Procedures 602 sharing it. As will be explained in detail later in the present embodiment, if a calling Procedure 602 does not share a PED 30303 with called Procedure 602 the Call must be mediated. A calling Procedure 602 may make a Neighborhood Call only to Procedures 602 with which it shares a PED 30303.

The next portion of Procedure Object 608 which is of interest is Internal Entry Descriptors 10342. Each Procedure 602 contained in Procedure Object 608 has an Entry Descriptor 47227. Entry Descriptor 47227 contains four fields of interest:

- PBP Offset Field 47229 contains the offset from PBP at which the first SIN in Procedure 602's code is located.
- Flags Field 47230 contains flags which are checked when Procedure 602 is invoked. Four flags are of interest:
 - Argument Information Array Present Flag 47235 which is set to TRUE if Procedure 602 has entries in Argument Information Array 10352.
 - SEB Flag 47237 is set to TRUE if SEPP 47225 is non-null, i.e., if Procedure 602 has a SEB 46864 for its S-Interpreter.
 - Do Not Check Access Flag 47239 is set to TRUE if KOS Call microcode is not to perform protection checking on the actual arguments used to invoke Procedure 602.
- PED Offset Field 47231 contains the offset of Procedure 602's PED 30303 from the beginning of Procedure Object 608.
- Frame Size Field 47233 contains the initial size of the Local Storage Portion 10420 of MAS Frame 46709 for an invocation of procedure 602.

Other areas of interest for Calls are SEB Prototype Area 47241, Static Data Area Prototype 30317, Binder Area 30323 and Argument Information Array 10352. SEB Prototype type Area 47241 and Static Data Area Prototype 30315 contain information used to create an SEB 46864 and Static Data Block 46863 respectively for Procedure 602. These areas are created on a per-MAS Object 46703 basis. The first time that a Process 610 executes a Procedure 602 in a domain, SEB 46864 and Static Data Block 46863 required for Procedure 602 are created either in MAS Object 46703 belonging to the domain or in another object accessible from MAS Object 46703. SEB 46864 and Static Data Block 46863 then remain as long as MAS Object 46703 exists.

Static Data Prototype 30317 contains two kinds of information: Static Data Links 30319 and Static Data Initialization Information 30321. Static Data Links 30319 contain locations in Binder Area 30323, which in turn contains pointers which may be resolved to yield the locations of data or External Procedures 602. When a Static Data Block 46863 is created for a Procedure 602, the information in Binder Area 30323 is used to create Linkage Pointers 46865. Static Data Initialization Information 30321 contains information required to create and initialize static data in Static Data Storage 46867.

As mentioned in the discussions of Link Gates 47206 and Static Data Links 30319 Binder Area 30323 contains pointers which may be resolved as described in Chapter 3 to yield locations of data and External Procedures 602.

Argument Information Array (AIA) 10352 contains information used by KOS Call microcode to check whether the subject which is invoking Procedure 602 has access to the actual arguments used in the invocation which allows the uses made of the arguments in Procedure 602. This so-called "Trojan horse check" is necessary because a Call may change the domain component of a subject. Thus, a subject which is lacking access of a specific kind to a data item could gain that access by passing the data item as an argument to a Procedure 602 whose DOE gives it access rights that the calling subject itself lacks.

Each Local Gate 47205 in Procedure Object 608 has an element in AIA 10352. Each of these Argument Information Array Elements (AIAEs) 60845 has fields indicating the following:

- The minimum number of arguments required to invoke Procedure 602 to which Local Gate 47205 belongs, in Field 47247.
- The maximum number of arguments which may be used to invoke Procedure 602 in Field 47249.
- The access rights that the invoking subject must have to the actual arguments in order to invoke Procedure 602 in Field 47251.

Field 47251 is itself an array which specifies the kinds of access that the invoking subject must have to the actual arguments it uses to invoke Procedure 602. Each formal argument for Procedure 602 has an Access Mode Array Entry (AMAE) 47255. The order of the AMAEs 47255 corresponds to the order of Procedure 602's formal arguments. The first formal argument has the first AMAE 47255, the second the second, and so forth. An AMAE 47253 is four bits long. There are two forms of AMAE 47253: Primitive Access Form 47255 and Extended Access Form 47257. In the former form, the leftmost bit is set to 0. The three remaining bits specify read, write, and execute access. If a bit is on, the subject performing the invocation must have that kind of primitive access to the object containing the data item used as an actual for the formal argument corresponding to that AMAE 47253. In the Extended Access Form 47257, the leftmost bit is set to 1 and the remaining bits are defined to represent extended access required for Procedure 602. The definition of these bits varies from Procedure 602 to Procedure 602.

5. Execution of Mediated Calls

Having described the portions of MAS Object 46703, SS Object 10336, and Procedure Object 608 which are involved in Calls, the discussion turns to the description of the Mediated Call Operation. First, there is presented an overview of the Mediated Call SIN and then the implementation of Mediated Calls in the present embodiment is discussed, beginning with a simple Mediated Call and continuing with Cross-Procedure Object Calls and Cross Domain Calls. The discussion closes with a description of software-to-microcode Calls.

a.a. Mediated Call SINS

While the exact form of a Mediated Call SIN is S-language specific, all Mediated Call SINS must contain four items of information:

- The SOP for the operation.
 - A Name that evaluates to a pointer to the Procedure 602 to be invoked by the SIN.
 - A literal (constant) specifying the number of actual arguments used in the invocation.
 - A list of Names which evaluate to pointers to the actual arguments used in the invocation.
- If Procedure 602 requires no arguments, the literal will be 0 and the list of Names representing the actual arguments will be empty.

In the present embodiment, Mediated Call and Return SINS are used whenever called Procedure 602 has a different PED 30303 from calling Procedure 602. In this case, the Call must save and recalculate macrostate other than FP and PC, and mediation by KOS Call microcode is required. The manner in which KOS Call microcode mediates the Call depends on whether the Call is a simple Mediated Call a Cross-procedure Object Call, or a Cross-Domain Call.

b.b. Simple Mediated Calls (Fig. 270, 468, 469, 470, 471, 472)

When the Mediated Call SIN is executed, S-interpreter microcode first evaluates the Name which represents the location of the called Procedure 602. The Name may evaluate to a pointer to a Gate 47205 or 4707 in another Procedure Object 608 or to a pointer to an Entry Descriptor 47227 in the present Procedure Object 608. When the Name has been evaluated, S-interpreter Call microcode invokes KOS Call microcode, using the evaluated Name as an argument. This microcode first fills in Macrostate Fields 10516, left empty until now, in the current invocation's SS Frame 47003. The microcode obtains the values for these fields from registers in FU 10120 where they are maintained while Virtual Processor 612 of Process 610 which is executing the Mediated Call is bound to JP 10114.

The next step to determine whether the pointer which KOS Call microcode received from S-interpreter Call microcode is a pointer to an External Procedure. To make this determination, KOS Call microcode compares the pointer's AON 41304 with that of Procedure Object 608 for Procedure 602 making the Call. If they are different, the Call is a Cross-Procedure Object Call, described below. In the case of the Simple Mediated Call, the format field indicates that the location is an Entry Descriptor 47227. KOS Call microcode continues by saving the location of Entry Descriptor 47227 and creating a new Mediated Frame 46947 on current MAS Object 46703 and a new Ordinary SS Frame 10510 on SS Object 10336 for called Procedure 602. As KOS Call microcode does so, it sets Fields 46917 and 46919 in Mediated Frame Header 10414 and

Fields 47109 and 47111 in Ordinary SS Frame Header 10514 to the values required by the addition of frames to MAS Object 46703 and SS Object 10336.

New Mediated Frame 46947 is now ready for Linkage Pointers 10416 to the actual arguments used in the Call, so KOS Call microcode returns to S-Interpreter Call microcode, which parses the SIN to obtain the literal specifying the number of arguments and saves the literal value. S-Interpreter Call microcode then parses each argument Name, evaluates it, and places the resulting value in Linkage Pointers Section 10416. When Linkage Pointers Section 10416 is complete, S-Interpreter Call Microcode calculates the new location of FP from the location of the top of Linkage Pointers Section 10416 and places a pointer for the location in the FU 10120 register reserved for FP. At this time, S-Interpreter Call microcode also places the new location of the top of the stack in Stack Top Offset Field 46807.

S-Interpreter Call microcode then invokes KOS Call microcode to place the value of the literal specifying the number of arguments in MAS Frame Field 46929, to calculate the new value of FHP 46702 and place it in the FU 10120 register reserved for that value, and finally to obtain the state necessary to execute called Procedure 602 from called Procedure 602's Entry Descriptor 47227 and PED 30303. As previously stated, S-Interpreter Call microcode saved the location of Entry Descriptor 47227. Using this location, KOS Call Microcode obtains the size of the storage required for local data from Field 47233 and adds that amount of storage to the new MAS Frame 46709. Then KOS Call Microcode uses Field 47231 to locate PED 30303 for Procedure 602. PED 30303 contains the remainder of the necessary information about Procedure 602 and KOS Call microcode copies the location of PED 30303 into PED Pointer Field 46933 and then copies the values of K Field 30305, Last Name Field 30307, NTP Field 30311 and PBP Field 30315 into the relevant registers in FU 10120. KOS Call microcode next translates the pointer in SIP Field 30309 into a dialect number as explained in Chapter 3, and places it in register RDIAL 24212 of FU 10220 and thereupon derives SDP by resolving the pointer in SDPP Field 30313 and a pointer to SEB 46864 by resolving the pointer in SEPP Field 30316. Having performed these operations, KOS Call microcode returns to S-Interpreter Call microcode, which finishes the Call by obtaining a new PC, that is, resetting registers in I-stream Reader 27001 in FU 10120 so that the next SIN to be fetched will be the first SIN of called procedure 602 S-Interpreter Call microcode obtains the information required to change PC from Field 47229 in Entry Descriptor 47227 which contains the offset of the first SIN of called Procedure 602 from PBP.

In the present embodiment, some FU 10120 state produced by the Mediated Call SIN is retained on SS 504 throughout the duration of Procedure 602's invocation. The saved state allows Process 610 to reattempt the Mediated Call if the Call fails before the called Procedure 602 begins executing. When a Mediated Return SIN is executed, it resumes execution on the retained state from the CALL SINT. The Mediated Return is much simpler than the Call. Since all of the information required to resume execution of the invocation which performed the Call is contained in Macrostate 10516 in the calling invocation's SS Frame 47003, Return need only pop the called invocation's frames from current MAS Object 46703 and SS Object 10336, copy Macrostate 10516 47123 from the calling invocation's SS Frame 47003 into the proper FU 10120 registers, translate SIP Value 47141 into a dialect number, and resume executing the calling invocation. The pop operation involves nothing more than updating those pointers in MAS Object 46703 and SS Object 10336 which pointed to locations in the old topmost frame so that they now point to equivalent locations in the new topmost frame.

c.c. Invocations of Procedures 602 Requiring SEBs 46864 (Fig. 270, 468, 469, 470, 471, 472)

If a Procedure 602 requires a SEB 46864, this fact is indicated by Flag Field 47237 in Procedure 602's Entry Descriptor 47227. PED 30303 for such a Procedure 602 contains SEPP Field 47225, whose value is a non-resolvable pointer. The manner in which a SEB 46864 is created for Procedure 602 and SEPP field 47225 is translated into SEP, a pointer which contains the location of SEB 46864 and is saved as part of the invocation's macrostate on SS 10336, is similar to the manner in which a Static Data Block 46863 is created and the non-resolvable pointer contained in SDPP field 47225 is translated into SDP. The first time that a Procedure 602 requiring a SEB 46864 is invoked on a MAS Object 46703, a SEB 46864 is created for the Procedure 602 and an AATE 46857 is created which associates the nonresolvable pointer in SEPP field 47225 and the location of SEB 46864. That location is the value of SEP when the procedure is executing on MAS object 46703. On subsequent invocations of Procedure 602, AATE 46857 serves to translate the value in SEPP field 47225 into SEP.

d.d. Cross-Procedure Object Calls (Fig. 270, 468, 469, 470, 471, 472)

A Mediated Call which invokes an External Procedure 602 is called a Cross-Procedure Object Call. As previously mentioned, KOS Call microcode assumes that any time the Name representing the called Procedure 602 in a Mediated Call SIN resolves to the location of a Gate that the Call is to an External Procedure 602. As long as newly-called External procedure 602 has the same DOE as calling Procedure 602. Cross-Procedure Object Calls differ from the Simple Mediated Call only in the manner in which called Procedure 602's Entry Descriptor 47227 is located. Once KOS Call microcode has determined as described above that a Mediated Call is a Cross-Procedure Object Call it must next determine whether it is a Cross-Domain Call. To do so, KOS Call microcode compares the DOE Attribute of called Procedure 602's Procedure Object 608 with the domain component of the current subject. KOS Call microcode uses Procedure Object 608's AON 41304 to obtain Procedure Object 608's DOE from Field 41521 of its AOTE

41306 and it uses the ASN for the current subject, stored in an FU 10120 register, to obtain the current subject's domain component from AST 10914. If the DOE and the current subject's domain component differ, the Call is a Cross-domain Call, described below; otherwise, the Call locates the Gate 47205 or 47206 specified by the evaluated Name for called Procedure 602 in its Procedure Object 608. If the Gate is a Local Gate 47205, the Call uses Entry Descriptor Offset Field 47207 to locate Entry Descriptor 47227 belonging to Called Procedure 602 and then proceeds as described in the discussion of a Simple Mediated Call.

If the Gate is a Link Gate 47206, KOS Call microcode obtains the pointer corresponding to Link Gate 47206 from Binder Area 47245 and resolves it to obtain a pointer to another Gate 47205 or 47206, which KOS Call microcode uses to repeat the External Procedure 602 call described above. The repetitions continue until the newly-located gate is a Local Gate 47205, whereupon Call proceeds as described for Simple Mediated Calls.

e.e. Cross-domain Calls (Fig. 270, 408, 418, 468, 469, 470, 471, 472)

If a called Procedure 602's Procedure Object 608 has a DOE attribute differing from that of calling Procedure 602's Procedure Object 608, the Call is a Cross-domain Call. The means by which KOS Call microcode determines that a Mediated Call is a Cross-Domain Call have previously been described; If the Call is a Cross-Domain Call, KOS Call microcode must inactivate MAS Object 46703 for the domain from which the Call is made, perform trojan horse argument checks, switch subjects, place a Cross-domain Frame 47039 on SS object 10336, and locate and activate MAS Object 46703 for the new domain before it can make a Mediated Frame 46947 on new MAS Object 46703 and continue as described in the discussion of a Simple Mediated Call.

Cross-domain Call microcode first inactivates the current MAS Object 46703 by setting Domain Active Flag 46804 to FALSE. The next step is the trojan horse argument checks. In order to perform trojan horse argument checks, Cross-domain Call must have pointers to the actual arguments used in the cross-domain invocation. Consequently, Cross-domain Call first continues like a non-cross-domain Call: it creates a Mediated Frame Header 10414 on old MAS Object 46703 and returns to S-interpreter microcode, which evaluates the Names of the actual arguments, and places the pointers in Linkage Pointers 10416 above Mediated Frame Header 10414. However, the macrostate for the invocation performing the call was placed on SS Object 10336 before Mediated Frame Header 10414 and Linkage Pointers 10416 were placed on old MAS Object 46703. Consequently, when calling Procedure 602 resumes execution after a Return, it will resume on MAS Frame 46709 preceding the one built by Cross-domain Call microcode.

Once the pointers to the actual arguments are available, Cross-domain Call Microcode performs the trojan horse check. As described in the discussion of Procedure Object 608 and illustrated in Figure 472, the information required to perform the check is contained in AIA 10352. Each Local Gate 47205 in Procedure Object 608 has an AIAE 47245, each formal argument in Local Gate 47205's procedure has an entry in AIAE 47245's AMA 47251, and the formal argument's AMAE 47253 indicates what kind of access to the formal argument's actual argument is required in called Procedure 602.

Field AIA OFF 47201 contains the location of AIA 10352 in Procedure Object 608, and using this information and Local Gate 47205's offset in Procedure Object 608, Cross-domain Call microcode locates AIAE 47245 for Local Gate 47205. The first two fields in AIAE 47245 contain the minimum number of arguments in the invocation and the maximum number of arguments. Cross-domain Call microcode checks whether the number of actual arguments falls between these values. If it does, Cross-domain Call microcode begins checking the access allowed individual arguments. For each argument pointer, Cross-domain Call microcode calls LAR microcode to obtain the current AON 41304 for the pointer's UID and uses AON 41304 and the ASN for Process 610's current subject (i.e., the caller's subject) to locate an entry in either APAM 10918 or ANPAT 10920, depending on whether the argument's AIAE specifies primitive access (47255) or extended access (47257) respectively. If the information from APAM 10918 or ANPAT 10920 confirms that Process 610's current subject has the right to access the argument in the manner required in called Procedure 602, the Trojan Horse microcode goes on to the next argument. If the current subject has the required access to all arguments, the trojan horse check succeeds and the Cross-domain Call continues. Otherwise, it fails and Cross-domain Call performs a microcode-to-software Call as explained below.

Next, Cross-domain Call microcode places Cross domain State 10513 on SS Object 10336. As explained in the discussion of SS object 10336, Cross-domain State 10513 contains the information required to return to the caller's frame on former MAS Object 46703. Having done this, Cross-domain Call microcode changes subjects. Using the current subject's ASN, Cross-Domain Call microcode obtains the current subject from AST 10914 replaces the subject's domain component with DOE Attribute 41225 for called Procedure 602's Procedure Object 608 and uses AST 10914 to translate the new subject thus obtained into a new ASN. That ASN then is placed in the appropriate FU 10120 register.

After the subject has been changed, Cross-domain Call microcode uses Domain Table 41801 to translate the DOE of called Procedure 602 into a domain number. Cross-domain Call microcode then uses the domain number as an index into Array of MAS AONs 46211 in VPSB 614 for Virtual Processor 612 belonging to Process 610 making the cross-domain call. The entry corresponding to the domain number contains AON 41304 of MAS Object 46703 for that domain.

Having located the proper MAS Object 46703, Cross-domain Call microcode uses STO field 46807 in MAS Header 10410 belonging to the new domains MAS Object 46703 to locate the top of the last MAS

Frame 46709. It then saves the value of FHP 46702 used in the preceding invocation in a FU 10120 register, adds a Mediated Frame Header 10414 to the top of MAS Object 46703, and calculates a new FHP 46702 which points to new Mediated Frame Header 10414. KOS Cross-Domain Call microcode then places the old value of FHP 46702 in FHP Value Field 47151 of SS Object 10336 and the old value of STO 46704 (pointing to the top of the last complete MAS Frame 46709 on previous MAS Object 46703) in Field 47153 of Cross-Domain State 10513 and fills in Mediated Frame Header 10414 fields as follows: Result of Cross-domain Call Field 46903 is set to TRUE. Previous Frame Offset Field 46917 is set to 0, and Dynamic Back Pointer Field 46931 is set to the saved value of FHP 46702. Dynamic Back Pointer Field 46931 thus points to the header of the topmost Mediated Frame 46947 on the previous MAS Object 46703. The values of the remaining fields are copied from Mediated Frame Header 10414 which Cross-Domain Call created on previous MAS Object 46703.

Cross-domain Call microcode next copies the argument pointers for the formal arguments from the top of previous MAS Object 46703 to new Mediated Frame 46947 and calculates FP. Cross-domain Call Microcode finishes by returning to S-Interpreter Call microcode, which completes the Call as described for Simple Mediated Calls.

Except for the work involved in transferring to a new MAS Object 46703, Cross-domain Return is like other Returns from Mediated Calls. Old FHP 46701 from Field 47151 of Cross-Domain State 10513 and old STO 46704, from Field 47153 of Cross-domain State are placed in FU 10120 registers. Then the frames belonging to the invocation that is ending are popped off of SS Object 10336 and off of MAS Object 46703 belonging to the domain of called Procedure 602 and MAS Object 46703 is inactivated by setting Domain-Active Flag 46804 to FALSE. Then KOS Cross-domain Return microcode uses old FHP 46701 and old STO 46704 to locate MAS Object 46703 being returned to and the topmost Mediated Frame 46947 on that MAS Object 46703. MAS Object 46703 being returned to is activated, and finally, the contents of Macrostate 10516 belonging to the invocation being returned to are placed in the appropriate registers of FU 10120 and execution of the invocation resumes.

f.f. Failed Cross-Domain Calls (Fig. 270, 468, 469, 470, 471, 472)

A Cross-Domain Call as described above may fail at several points between the time that the calling invocation begins the call and called Procedure 602 begins executing. On failure, Cross-Domain Call microcode performs a microcode-to-software Call. KOS Procedures 602 invoked by this Call may remedy the reason for the Cross Domain Call's failure and reattempt the Cross-domain Call. This is possible because the implementation of Cross Domain Call in CS 10110 saves sufficient FU 10120 state to allow Process 610 executing the Cross-Domain Call to return to the invocation and the Mediated Call SIN from which the Cross-Domain Call began. On failure, the invocation's MAS Frame 46709 may be located from the values of STO Field 47153 and FHP Field 47151 in Cross-Domain State 10513, and the Mediated Call SIN may be located by using information saved in FU 10120 state.

6. Neighborhood Calls (Fig. 468, 479, 472)

As previously mentioned, Procedures 602 called via Neighborhood Calls must have the same PED 30303 as calling Procedure 602. The only macrostate values which are not part of PED 30303 are PC and FP; consequently Neighborhood Call need only save PC and FP of the invocation performing the call and calculate these values for the new invocation. In addition, Neighborhood Call saves STO 46704 in order to make it easier to locate the top of the previous invocation's Neighborhood Frame 46947. Neighborhood Return simply restores the saved values. Since the macrostate values copied from or obtained via PED 30303 do not change during the sequence of invocations, and therefore need not be saved on SS Object 10336. Neighborhood Calls do not have SS Frames 47003.

The invention may be embodied in yet other specific forms without departing from the spirit or essential characteristics thereof. Thus, the present embodiments are to be considered in all respects as illustrative and not restrictive, the scope of the invention being indicated by the appended claims rather than by the foregoing description.

Claims

1. A digital computer system (CS 101) including processor means (JP 114) for performing operations upon operands, memory means (MEM 112) for storing said operands and procedures, said procedures including instructions for controlling said operations and names referring to certain of said operands to be operated upon, ALU means (2034, 2074) for performing said operations, bus means (MOD 140, JPB 142) for conducting said instructions, names and operands between said memory means and said processor means, and I/O means (IOS 116) for conducting at least said operands between said memory means and devices external to said digital computer system, characterised in that said processor means (JP 114) comprises means for addressing said operands, including name table means (10350) for storing name table entries, each name table entry corresponding to one of said names included in each one of said procedures and each name table entry comprising first data from which may be determined an address of a location in said memory means of the operand referred to by one of said names and second data identifying a format of that operand, and translation means (NAME TRANS UNIT 27015) connected to said

bus means and responsive to said name table entries for providing outputs to said memory means representing said addresses, and further characterised in that said instructions are intermediate level S-language instructions from a plurality of sets of such instructions, each set corresponding to a particular higher level user programming language, and further characterised by receiving means (INSTB 20262) connected to said bus means for receiving said instructions from said memory means, and microcode control means (10240, 27003, 27013) connected between said receiving means and said ALU means for providing sequences of microinstructions for controlling said ALU means, said sequences being selected from a plurality of sequences of microinstructions corresponding to said S-language instructions respectively.

2. A digital computer system according to claim 1, characterised in that the S-language instructions have a uniform, fixed format.

3. A digital computer system according to claim 1 or 2, characterised in that the names are of uniform length and format.

4. A digital computer system according to any of claims 1 to 3, characterised in that each procedure further includes a name table pointer (NTP 30311) representing a base location in said memory means (MEM 112), and said first data of each name table entry contains information from which may be determined an address offset of a memory location relative to the base location, and in that said translation means (NAME TRANS UNIT 27015) further comprises base register means (NCR, MCR 10366) connected to said bus means for receiving and storing said name table pointer of the procedure currently controlling the operations performed by said ALU means.

5. A digital computer system according to any of claims 1 to 4, characterised by name cache means (10226) connected to outputs of said translation means (NAME TRANS UNIT 27015) and having outputs to said memory means (MEM 112) for storing said addresses, and further connected to said receiving means (INSTB 20262) and responsive to said names to provide name cache outputs to said memory means representing said addresses of certain operands for which said name cache means has stored said addresses.

6. A digital computer system according to any of claims 1 to 5, characterised in that each of said S-Language instructions is a member of an S-Language dialect of a plurality of S-Language dialects, and in that said receiving means (INSTB 20262) further comprises dialect code means (RDIAL 24212) for storing a dialect code specifying the dialect of which the received S-Language instructions are members, and in that said sequences of microinstructions include a set of sequences of microinstructions, corresponding to each said S-Language dialect, each set of sequences of microinstructions including at least one sequence of microinstructions corresponding to each S-Language instruction in a corresponding S-Language dialect, and in that said microcode control means (10240, 27003, 27013) is responsive to the dialect code and to each received S-Language instruction to provide to said ALU means (2034, 2074) a sequence of microinstructions corresponding to that S-Language instruction.

7. A digital computer system according to claim 1 or 2, characterised in that each procedure includes a dialect code denoting an S-Language dialect of which the S-Language instructions of the procedure are members, and in that said microcode control means (10240, 27003, 27013) further comprises control store means (SITT 11012) for storing said sequences of microinstructions for controlling said ALU means (2034, 2074), and dispatch table means (SIDT 11010) for storing addresses corresponding to locations in said control store means of each sequence of microinstructions, and in that said dispatch table means is responsive to said dialect code and to each instruction to provide to said control store means each address corresponding to said at least one microinstruction sequence corresponding to each said instruction, and said control store means is responsive to each address to provide to said ALU means said sequence of microinstructions corresponding to each instruction.

8. A digital computer system according to claim 1, 6 or 7, characterised in that said microcode control means (10240, 27003, 27013) comprises writable control store means (11012) connected to said bus means for storing said sequences of microinstructions, and control store addressing means (SITNAS 20286) responsive to each S-Language instruction and to operation of said processor means for generating control store read addresses and write addresses (CSADR 20204), and in that said writable control store means is responsive to said read addresses to provide said sequences of microinstructions to said ALU means (2034, 2074) and is responsive to said write addresses to store said sequences of microinstructions.

9. A digital computer system according to claim 7, characterised in that said control store means (SITT 11012) comprises writable control store means connected to said bus means for storing said sequences of microinstructions, and in that said dispatch table means comprises write address means responsive to operation of said processor means for generating write addresses, and in that said writable control store means is responsive to said write addresses for storing said sequences of microinstructions.

60 Patentansprüche

1. Digitales Datenverarbeitungssystem (CS 101), enthaltend: Prozessormittel (MEM 114) zur Durchführung von Operationen an Operanden, Speichermittel (MEM 112) zum Speichern der Operanden und von Prozeduren, die Befehle zur Steuerung der Operationen und Namen enthalten, die auf gewisse der Operanden Bezug nehmen, an denen Operationen durchgeführt werden sollen, ein Rechenwerk (2034,

2074) zur Durchführung der Operationen, Bus-Mittel (MOD 140, JPE 118) für den Verkehr der Befehle, Namen und Operanden zwischen den Speichermitteln und den Prozessormitteln, und Eingabe/Ausgabemittel (IOS 116) für den Verkehr wenigstens der Operanden zwischen den Speichermitteln und Geräten außerhalb des digitalen Datenverarbeitungssystems, gekennzeichnet durch Prozessormittel (JP 114), die Mittel zur Adressierung der Operanden einschließlich Namenstabellenmittel (10350) zur Speicherung von Namenstabellen-Einsprungpunkten enthalten, wobei jeder Namenstabellen-Einsprungpunkt einem der Namen entspricht, die in jeder der Prozeduren enthalten sind, und erste Daten, aus denen eine Adresse eines Platzes derjenigen Operanden in den Speichermitteln bestimmt werden kann, auf die durch einen der Namen Bezug genommen wird, und zweite Daten enthalten die ein Format dieses Operanden identifizieren, und durch Übersetzungsmittel (NAME TRANS UNIT 27015), die mit den Bus-Mitteln verbunden sind und auf die Namenstabellen-Einsprungpunkte unter Bereitstellung von diese Adressen repräsentierenden Ausgaben für die Speichermittel ansprechen, ferner dadurch gekennzeichnet, daß die Befehle mittlere S-Sprache-Befehle von einer Vielzahl von Sätzen solcher Befehle sind, von denen jeder Satz einer besonderen höheren Benutzerprogrammiersprache entspricht, und ferner gekennzeichnet durch ein mit den Bus-Mitteln verbundenes Empfangsmittel (INSTB 20262) zum Empfang der Befehle von den Speichermitteln, und durch mit dem Empfangsmittel und dem Rechenwerk verbundene Mikrocode-Steuermittel (10240, 27003, 27013) zur Bereitstellung von Mikrobefehlssequenzen zur Steuerung des Rechenwerks, wobei diese Sequenzen aus einer Vielzahl von Mikrobefehlssequenzen ausgewählt sind, die den jeweiligen S-Sprache-Befehlen entsprechen.

2. Digitales Datenverarbeitungssystem nach Anspruch 1, dadurch gekennzeichnet, daß die S-Sprache-Befehle ein gleichförmiges, festes Format haben.

3. Digitales Datenverarbeitungssystem nach Anspruch 1 oder 2, dadurch gekennzeichnet, daß die Namen eine gleichförmige Länge und ein gleichförmiges Format haben.

4. Digitales Datenverarbeitungssystem nach einem der Ansprüche 1 bis 3, dadurch gekennzeichnet, daß jede Prozedur weiter einen Namenstabellenzeiger (NTP 30311) enthält, der einen Basisplatz in den Speichermitteln (MEM 112) repräsentiert, daß die ersten Daten jedes Namenstabellen-Einsprungpunktes Informationen enthalten, aus denen die Adresse eines vom Basispeicherplatz versetzten Speicherplatzes bestimmt werden können, und daß die Übersetzungsmittel (NAME TRANS UNIT 27015) weiter Basisregistermittel (NCR, MCR 10366) enthalten, die mit den Bus-Mitteln verbunden sind, um den Namenstabellenzeiger derjenigen Prozedur zu empfangen und zu speichern, die gerade die vom Rechenwerk durchgeführten Operationen steuert.

5. Digitales Datenverarbeitungssystem nach einem der Ansprüche 1 bis 4, gekennzeichnet durch Namens-Cache-Speichermittel (10226), die mit den Ausgängen der Übersetzungsmittel (NAME TRANS UNIT 27015) verbunden sind und zu den Speichermitteln (MEM 112) führend Ausgänge zum Speichern der Adressen haben, und die weiter mit dem Empfangsmittel (INSTB 20262) verbunden sind und auf die Namen unter Bereitstellung von Namens-Cache-Ausgaben für die Speichermittel ansprechen, die die Adressen von gewissen Operanden repräsentieren, für die die Namens-Cache-Speichermittel die Adressen gespeichert haben.

6. Digitales Datenverarbeitungssystem nach einem der Ansprüche 1 bis 5, dadurch gekennzeichnet, daß jeder der S-Sprache-Befehle ein Mitglied eines S-Sprache-Dialekts einer Vielzahl von S-Sprache-Dialekten ist, daß das Empfangsmittel (INSTB 20262) weiter ein Dialekt-Code-Mittel (RDIAL 24212) zur Speicherung eines Dialekt-Codes enthält, der den Dialekt bestimmt, von dem die empfangenen S-Sprache-Befehle Mitglieder sind, daß die Mikrobefehlssequenzen einen Satz von Mikrobefehlssequenzen entsprechend jedem S-Sprache-Dialekt enthalten, wobei jede Mikrobefehlssequenz wenigstens eine jedem S-Sprache-Befehl in einem entsprechenden S-Sprache-Dialekt entsprechenden Mikrobefehlssequenz enthält, und daß die Mikrocode-Steuermittel (10240, 27003, 27013) auf den Dialekt-Code und jeden empfangenen S-Sprache-Befehl unter Bereitstellung einer diesem S-Sprache-Befehl entsprechenden Mikrobefehlssequenz für das Rechenwerk ansprechen.

7. Digitales Datenverarbeitungssystem nach Anspruch 1 oder 2, dadurch gekennzeichnet, daß jede Prozedur einen Dialektcode enthält, der einen S-Sprache-Dialekt bezeichnet, von dem die S-Sprache Befehle der Prozedur Mitglieder sind, daß die Mikrocode-Steuermittel (10240, 27003, 27013) ferner Speichermittel (SITT 11012) zur Speicherung der Mikrobefehlssequenzen für die Steuerung des Rechenwerks (2034, 2074) und Verteilertabellenmittel (SIDT 11010) zur Speicherung von Adressen enthalten, die Plätzen jeder Mikrobefehlssequenz in den Speichermitteln entsprechen, und daß die Verteilertabellenmittel auf den Dialektcode und jeden Befehl unter Bereitstellung jeder Adresse, die der wenigstens einen, zu jedem Befehl gehörenden Mikrobefehlssequenz entspricht, für die Speichermittel ansprechen, während die Speichermittel auf jede Adresse unter Bereitstellung der jedem Befehl entsprechenden Mikrobefehlssequenz für das Rechenwerk ansprechen.

8. Digitales Datenverarbeitungssystem nach Anspruch 1, 6 oder 7, dadurch gekennzeichnet, daß die Mikrocode-Steuermittel (10240, 27003, 27013) ein mit den Bus-Mitteln verbundenes Schreibsteuerspeichermittel (11012) zur Speicherung der Mikrobefehlssequenzen und Steuerspeicheradressiermittel (SITNAS 20286) enthalten, die auf jeden S-Sprache-Befehl und auf Operationen des Prozessormittels unter Erzeugung von Steuerspeicherlese- und -schreibadressen (CSADR 20204) ansprechen, und daß die Schreibsteuerspeichermittel auf die Leseadressen unter Bereitstellung der Mikrobefehlssequenzen für das Rechenwerk und auf die Schreibadressen unter Speicherung dieser Mikrobefehlssequenzen ansprechen.

EP 0 067 556 B1

9. Digitales Datenverarbeitungssystem nach Anspruch 7, dadurch gekennzeichnet, daß die Steuerspeichermittel (SITT 11012) mit den Bus-Mitteln verbundene Schreibsteuerspeichermittel zur Speicherung der Mikrobefehlssequenzen enthalten, daß die Verteilertabellenmittel Schreibadressenmittel enthalten, die auf Operationen des Prozessormittels unter Erzeugung von Schreibadressen ansprechen, und daß die Schreibsteuerspeichermittel auf die Schreibadressen unter Speicherung der Mikrobefehlssequenzen ansprechen.

Revendications

1. Un système d'ordinateur numérique (CS 101), comprenant un processeur (JP 114) pour effectuer des opérations sur des opérandes, une mémoire (MEM 112) pour mémoriser lesdits opérandes et des procédures, lesdites procédures contenant des instructions pour commander lesdites opérations et des désignations se rapportant à certains desdits opérandes pour les traiter, une unité arithmétique et logique ALU (2034, 2074) pour effectuer lesdites opérations, des bus (MOD 140, JPB 142) pour transmettre lesdites instructions, lesdites désignations et lesdits opérandes entre ladite mémoire et ledit processeur, et des moyens d'entrée/sortie I/O (IOS 116) pour transmettre au moins lesdits opérandes entre ladite mémoire et des dispositifs extérieurs audit système d'ordinateur numérique, caractérisé en ce que ledit processeur (JP 114) comprend des moyens pour l'adressage desdits opérandes, comportant une table de désignations (10350) pour mémoriser des entrées de table de désignations, chaque entrée de table de désignations correspondant à une desdites désignations incluses dans chacune desdites procédures et chaque entrée de table de désignations comprenant une première donnée à partir de laquelle peut être déterminée une adresse d'un emplacement de ladite mémoire contenant l'opérande auquel se reflète l'une desdites désignations et une seconde donnée identifiant un format de cet opérande, et des moyens de transcodage (NAME TRANS UNIT 27015) reliés auxdits bus et réagissant auxdites entrées de tables de désignations de façon à transmettre à ladite mémoire des signaux de sortie représentant lesdites adresses, et en outre caractérisé en ce que lesdites instructions sont des instructions en langage-S de niveau intermédiaire provenant d'une pluralité d'ensembles de telles instructions, chaque ensemble correspondant à un langage de programmation par utilisateur de niveau supérieur particulier, et en outre caractérisé en ce que des moyens de réception (INSTB 20262) sont reliés auxdits bus pour recevoir lesdites instructions à partir de ladite mémoire, et des moyens de commande de microcode (10240, 27003, 27013) connectés entre lesdits moyens de réception et ladite ALU pour fournir des séquences de microinstructions servant à commander ladite ALU, les dites séquences étant sélectionnées parmi une pluralité de séquences de micro-instructions correspondant respectivement auxdites instructions en langage-S.

2. Un système d'ordinateur numérique selon la revendication 1, caractérisé en ce que les instructions en langage-S ont un format fixe et uniforme.

3. Un système d'ordinateur numérique selon une des revendications 1 ou 2, caractérisé en ce que les désignations ont une longueur et un format uniformes.

4. Un système d'ordinateur numérique selon une quelconque des revendications 1 à 3, caractérisé en ce que chaque procédure comprend en outre un pointeur de table de désignations (NTP 30311) représentant un emplacement de base dans ladite mémoire (MEM 112) et ladite première donnée de chaque entrée de la table de désignations contient une information à partir de laquelle peut être déterminé un décalage d'adresse d'un emplacement de mémoire par rapport à l'emplacement de base, et en ce que lesdits moyens de transcodage (NAME TRANS UNIT 27015) comprennent en outre un moyen formant registre de base (NCR, MCR 10366), qui est relié auxdits bus de façon à recevoir et mémoriser ledit pointeur de table de désignations dans la procédure qui est en train de commander les opérations effectuées par ladite ALU.

5. Un système d'ordinateur numérique selon une quelconque des revendications 1 à 4, caractérisé par un moyen formant antémémoire de désignations (10226), relié aux sorties desdits moyens de transcodage (NAME TRANS UNIT 27015) et comportant des sorties reliées à ladite mémoire (MEM 112) pour mémoriser lesdites adresses, et en outre relié auxdits moyens de réception (INSTB 20262) et réagissant auxdites désignations pour fournir à ladite mémoire des sorties de l'antémémoire de désignations représentant lesdites adresses de certains opérandes pour lesquels ladite antémémoire de désignations a mémorisé lesdites adresses.

6. Un système d'ordinateur numérique selon une quelconque des revendications 1 à 5, caractérisé en ce que chacune desdites instructions en langage-S est un élément d'un dialecte en langage-S faisant partie d'une pluralité de dialectes en langage-S et en ce que lesdits moyens de réception (INSTB 20262) comprennent en outre un moyen de codage de dialecte (RDIAL 24212) pour mémoriser un code de dialecte spécifiant le dialecte dont les instructions en langage-S reçues sont des éléments, et en ce que lesdites séquences de micro-instructions contiennent un ensemble de séquences de micro-instructions correspondant à chacun desdits dialectes en langage-S, chaque ensemble de séquences de micro-instructions comprenant au moins une séquence de micro-instructions correspondant à chaque instruction en langage-S dans un dialecte en langage-S correspondant, et en ce que lesdits moyens de commande de microcode (10240, 27003, 27013) réagissent audit code de dialecte et à chaque instruction en langage-S reçue pour fournir à ladite ALU (2034, 2074) une séquence de micro-instructions correspondant à cette instruction en langage-S.

7. Un système d'ordinateur numérique selon une des revendications 1 et 2, caractérisé en ce que chaque procédure comprend un code de dialecte définissant un dialecte en langage-S dont les instructions en langage-S de la procédure sont des éléments et en ce que lesdits moyens de commande de microcode (1020, 27003, 27013 comprennent en outre une mémoire de commande (SITT 11012) pour mémoriser lesdites séquences de micro-instructions pour commander ladite ALU (2034, 2074), et un moyen à table de distribution (SIDT 11010) pour mémoriser des adresses correspondant aux emplacements de chaque séquence de micro-instructions dans ladite mémoire de commande, et en ce que ledit moyen à table de distribution réagit audit code de dialecte et à chaque instruction pour fournir à ladite mémoire de commande chaque adresse correspondant à ladite séquence de micro-instructions au moins prévue correspondant à chacune desdites instructions, et ladite mémoire de commande réagit à chaque adresse pour fournir à ladite ALU ladite séquence de micro-instructions correspondant à chaque instruction.

8. Un système à ordinateur numérique selon une des revendications 1, 6 et 7, caractérisé en ce que lesdits moyens de commande de microcode (10240, 27003, 27013) comprennent une mémoire de commande inscriptible (11012) reliée auxdits bus pour mémoriser lesdites séquences de micro-instructions et un moyen d'adressage de mémoire de commande (SITTNAS 20286) réagissant à chaque instruction en langage-S et au fonctionnement dudit processeur pour produire des adresses de lecture et des adresses d'écriture dans la mémoire de commande (CSADR 20204) et en ce que ladite mémoire de commande inscriptible réagit auxdites adresses de lecture pour fournir lesdites séquences de micro-instructions à ladite ALU (2034, 2074) et réagit auxdites adresses d'écriture pour mémoriser lesdites séquences de micro-instructions.

9. Un système d'ordinateur numérique selon la revendication 7, caractérisé en ce que ladite mémoire de commande (SITT 11012) comprend une mémoire de commande inscriptible qui est reliée auxdits bus de mémoriser lesdites séquences de micro-instructions et en ce que ledit moyen à table de distribution comprend un moyen d'adressage d'écriture réagissant au fonctionnement dudit processeur pour produire des adresses d'écriture, et en ce que la mémoire de commande inscriptible réagit auxdites adresses d'écriture pour mémoriser lesdites séquences de micro-instructions.

30

35

40

45

50

55

60

65

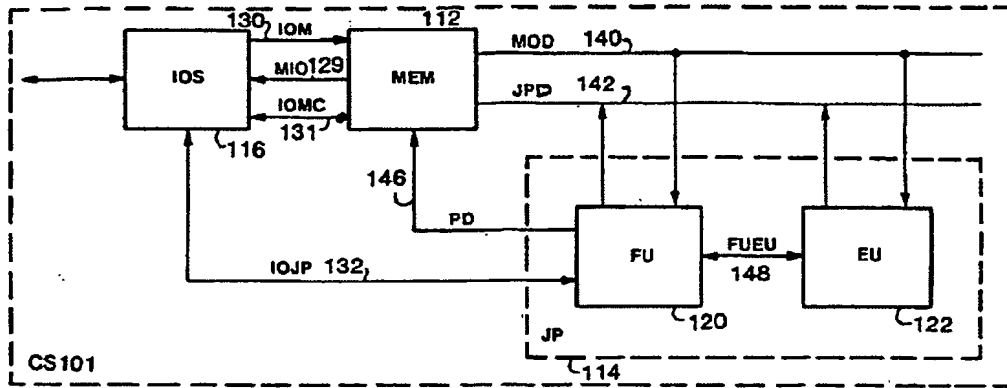


FIG 1

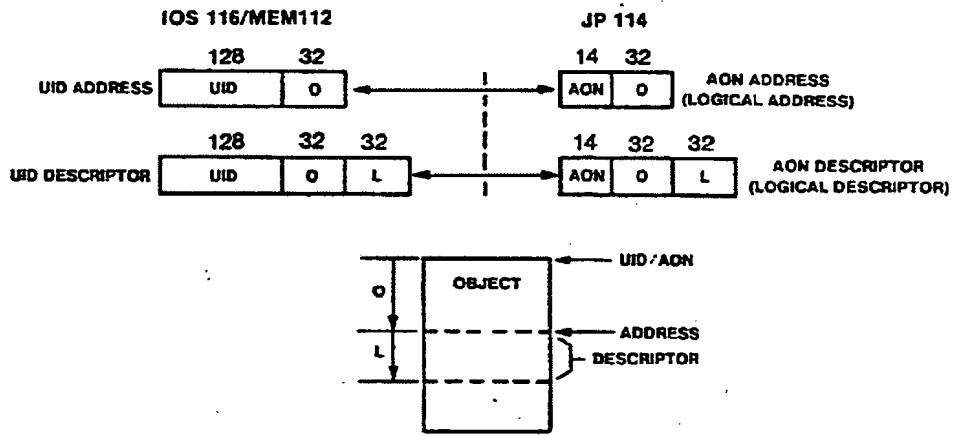


FIG 2

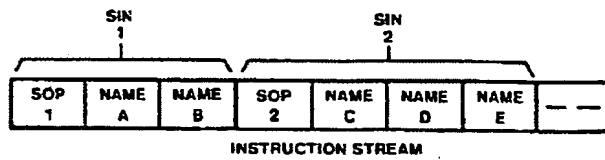


FIG 3

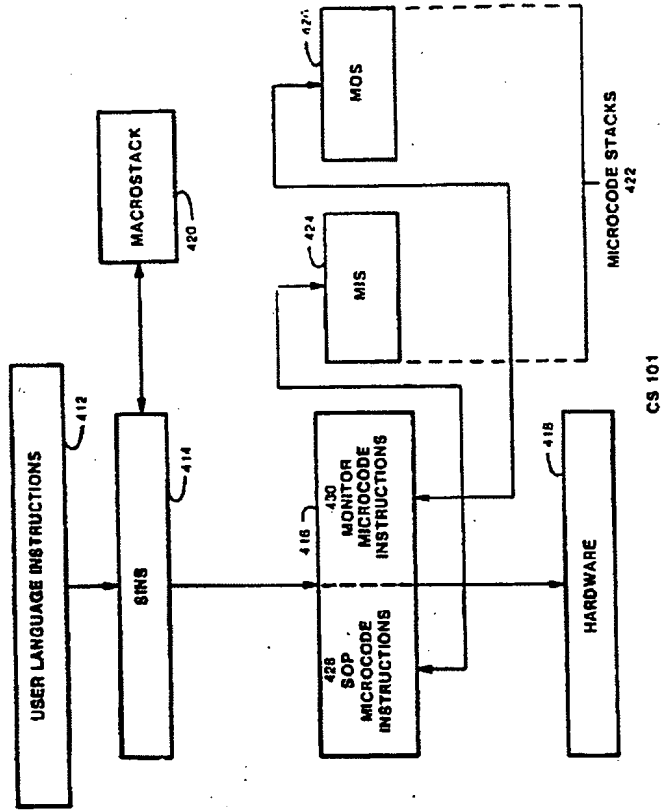


FIG 4A

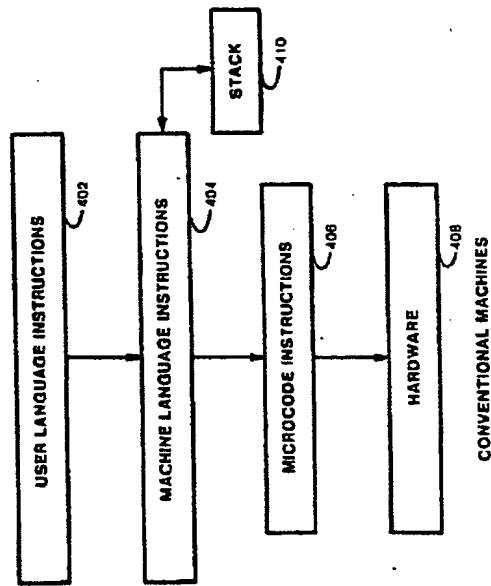


FIG 4

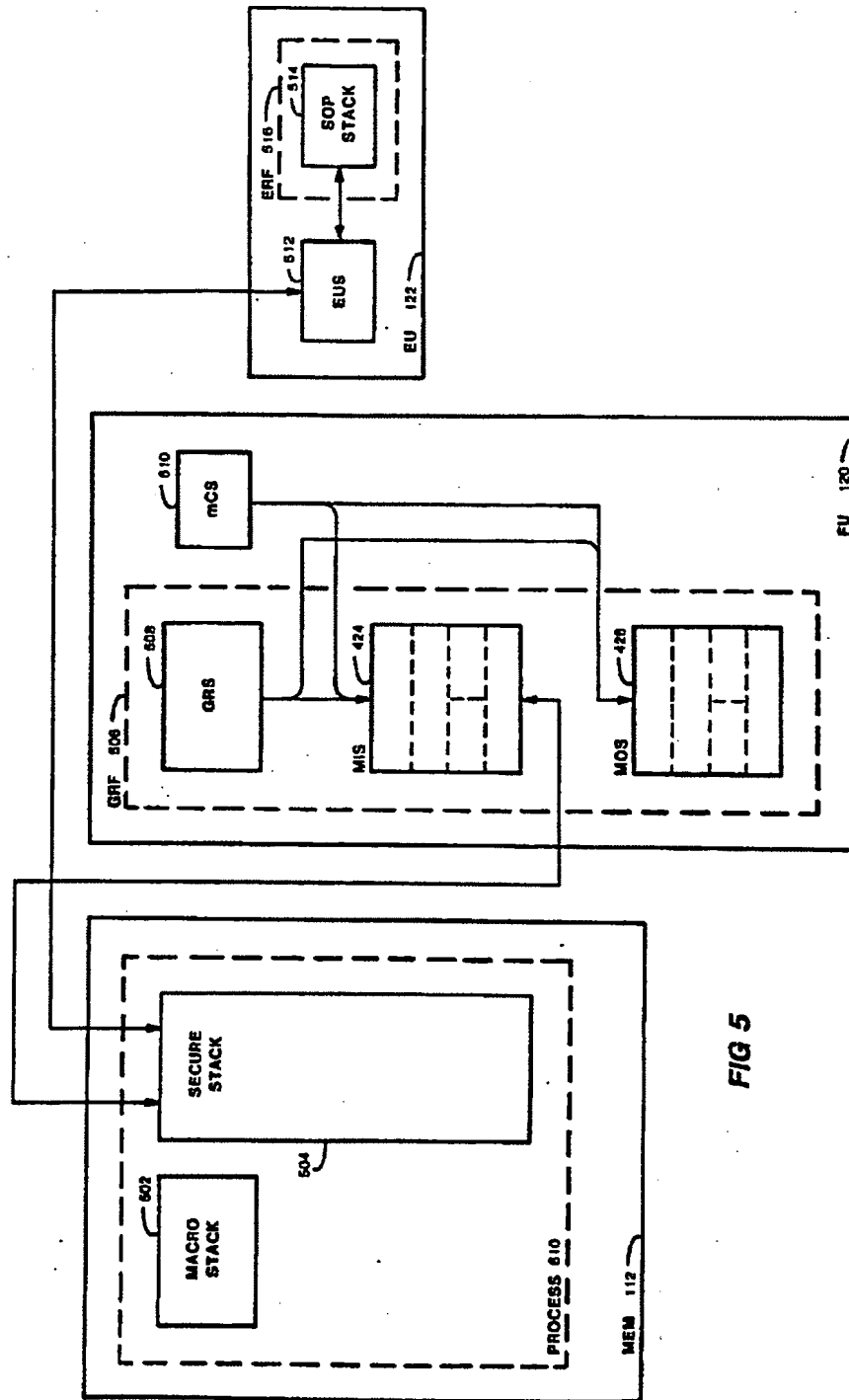


FIG 5

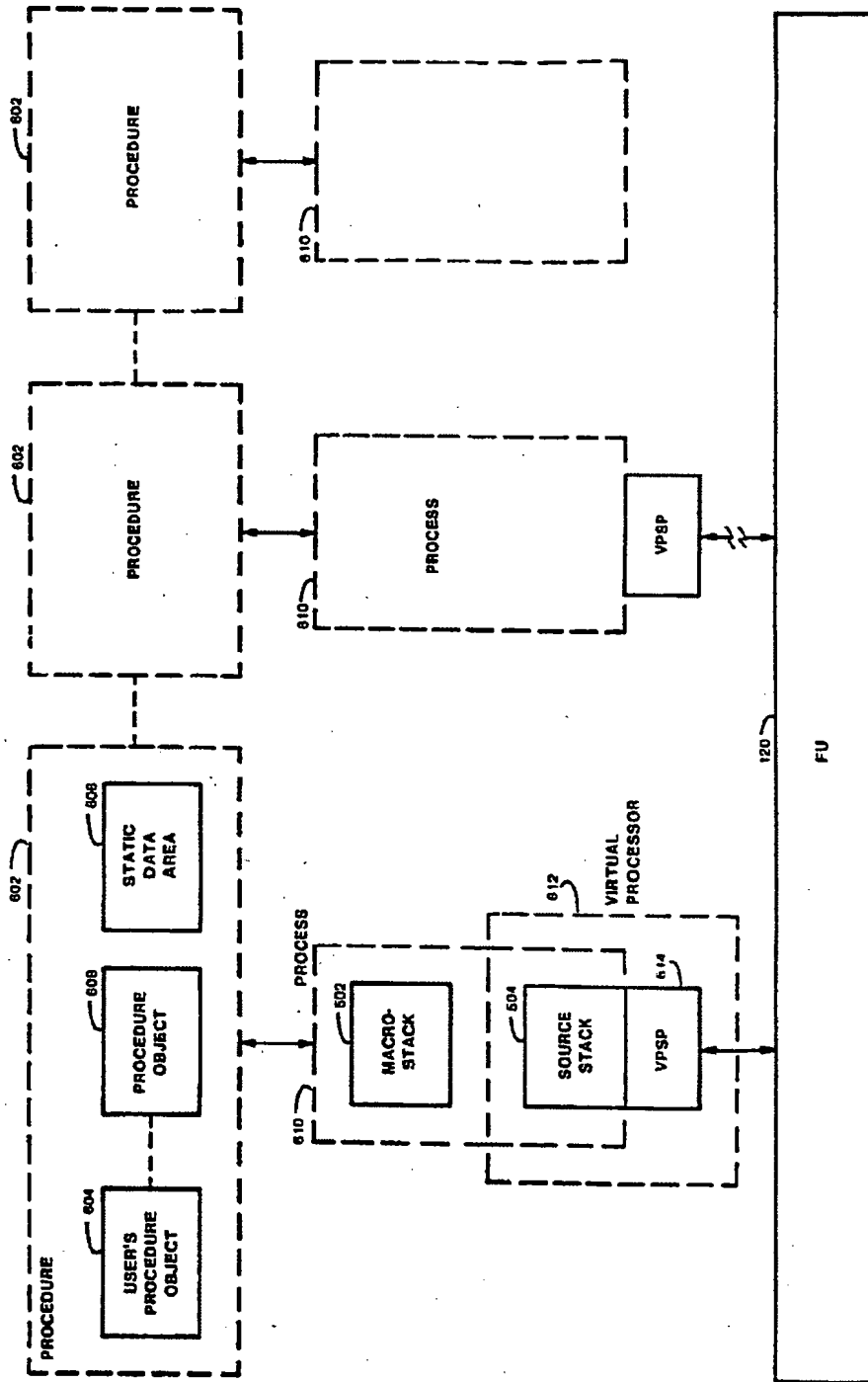


FIG 6

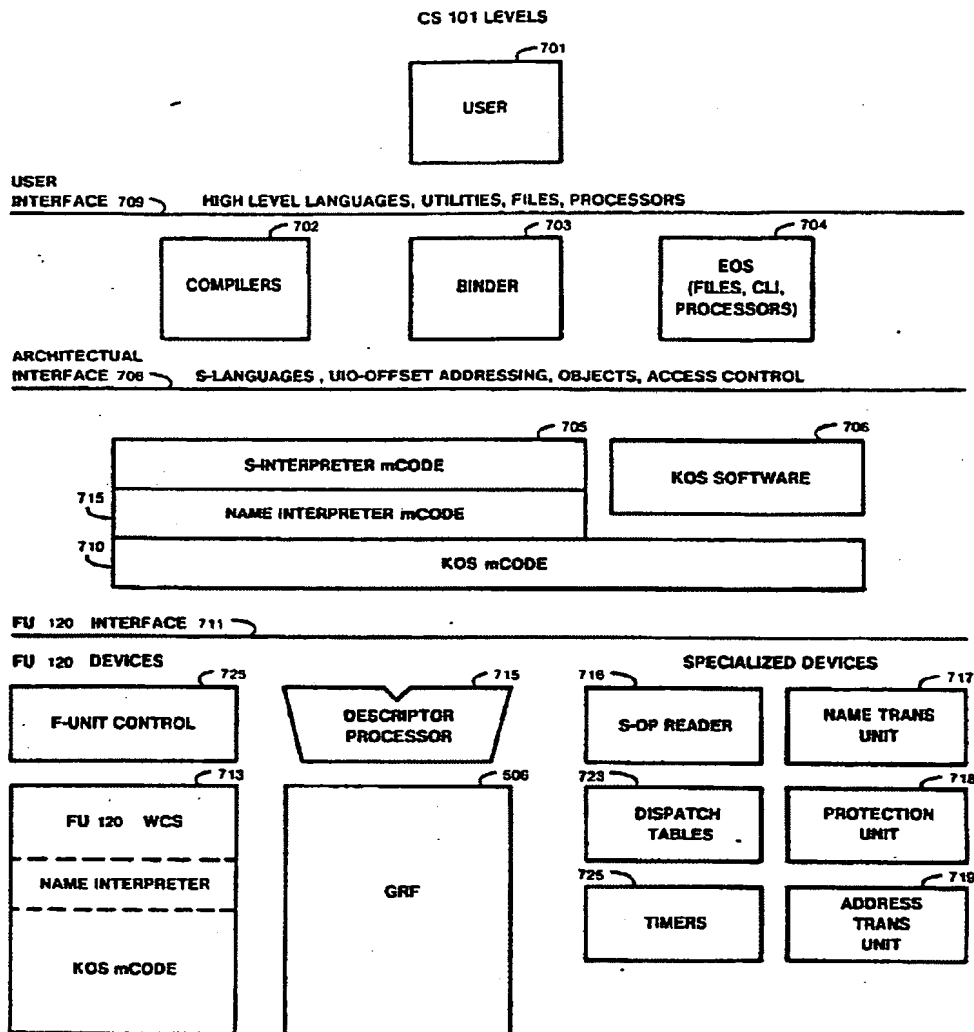


FIG 7

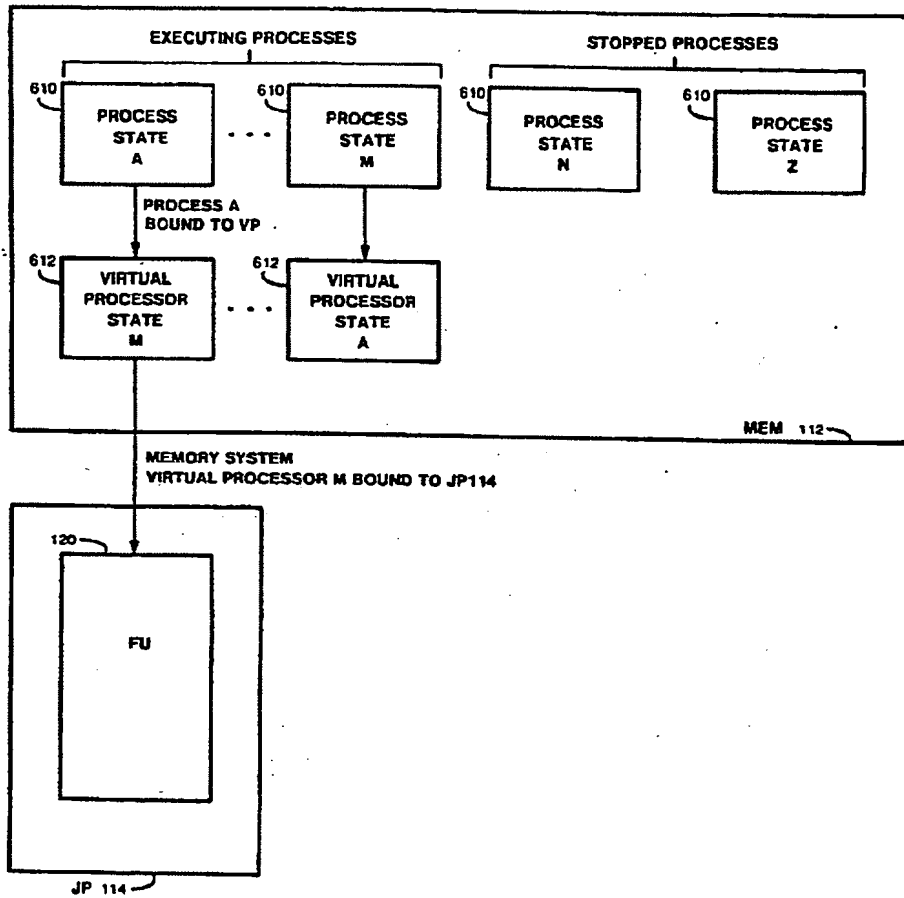


FIG 8

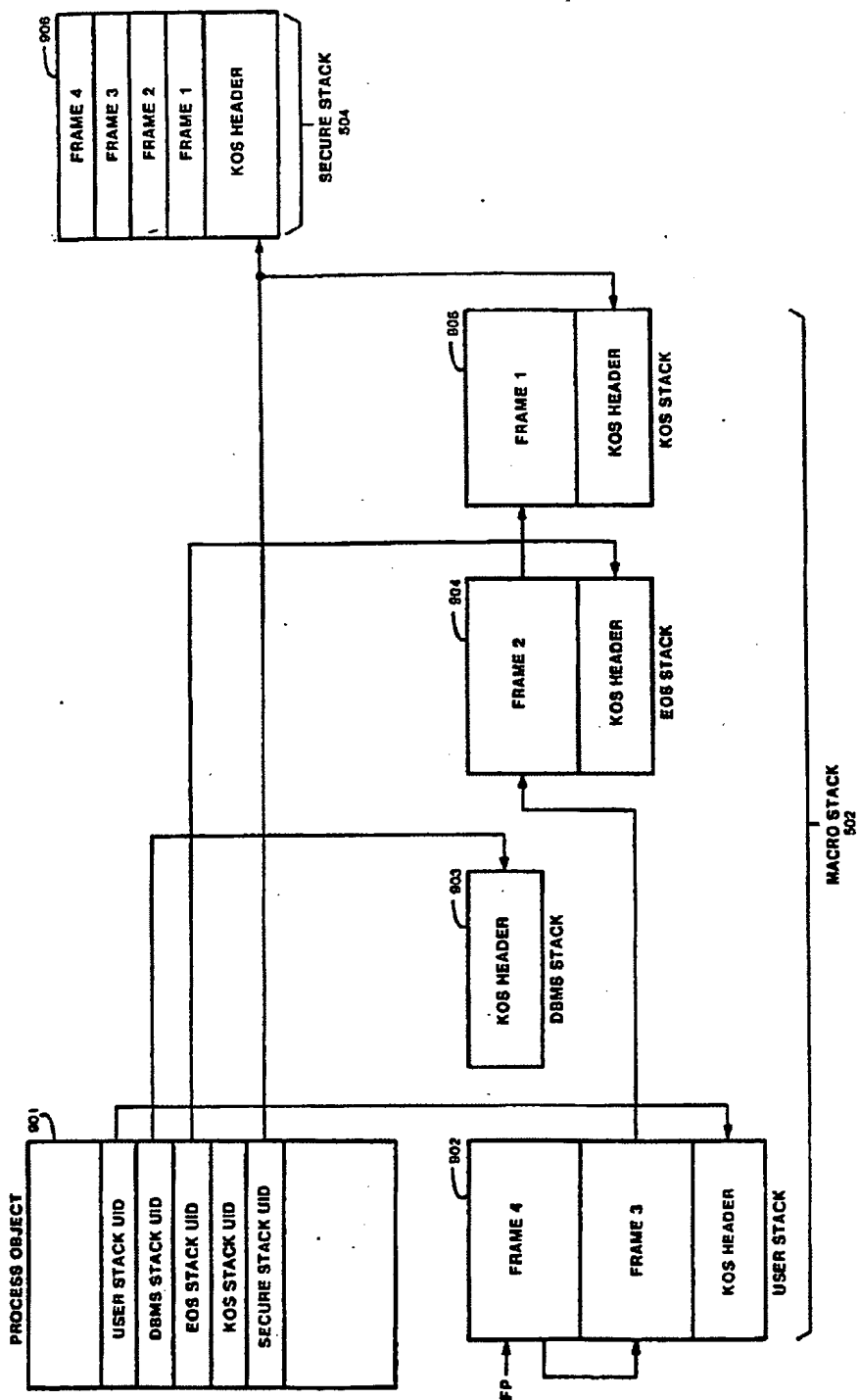


FIG 9

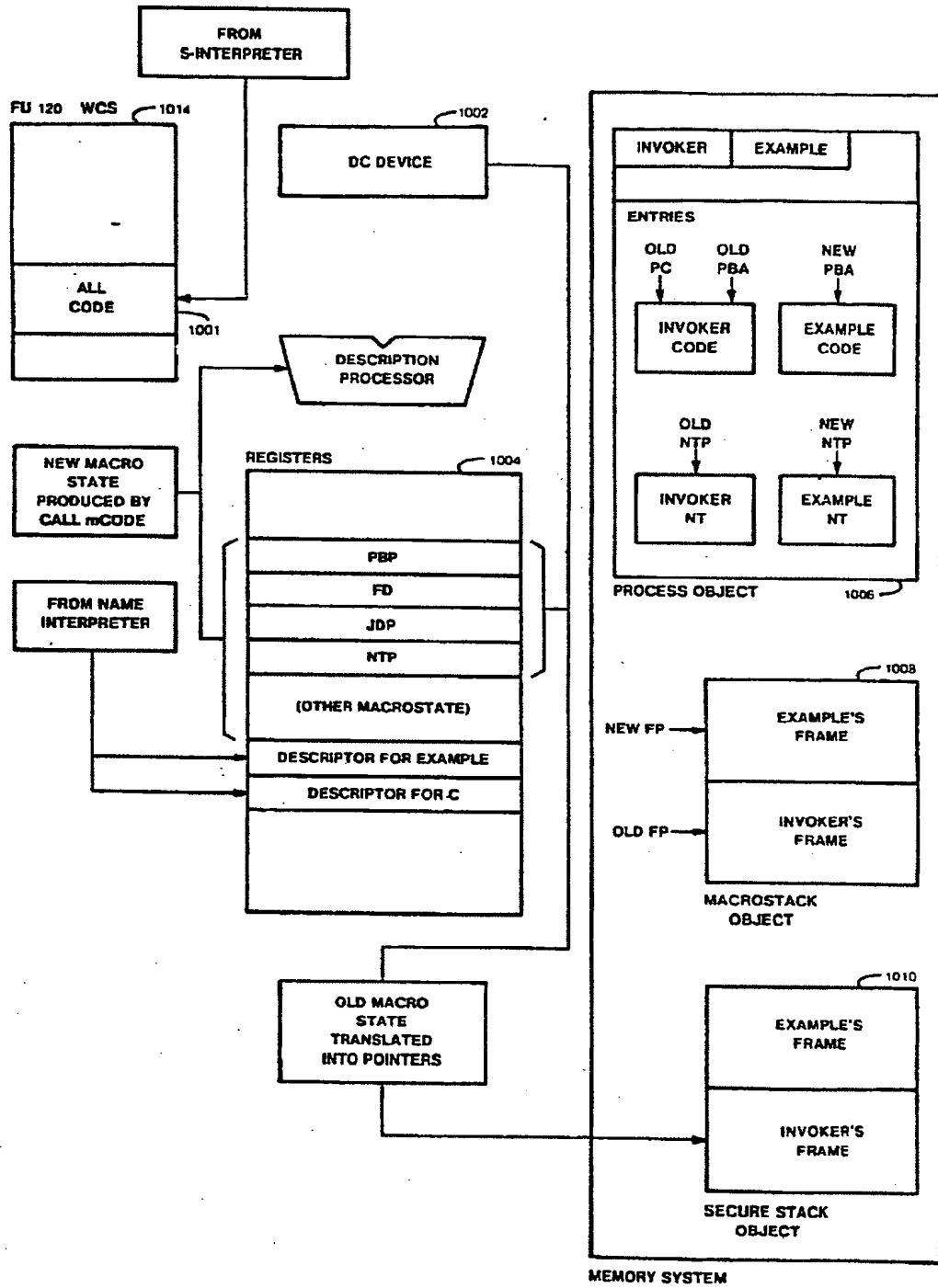


FIG 10

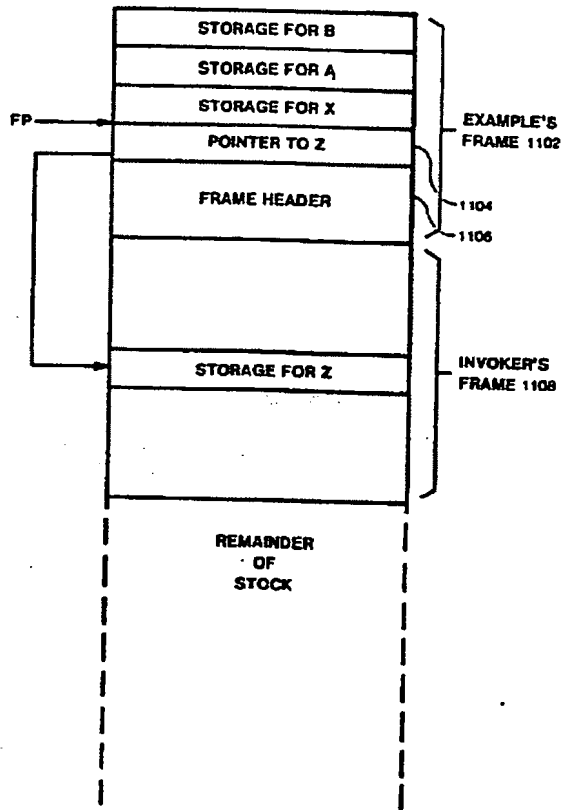


FIG 11

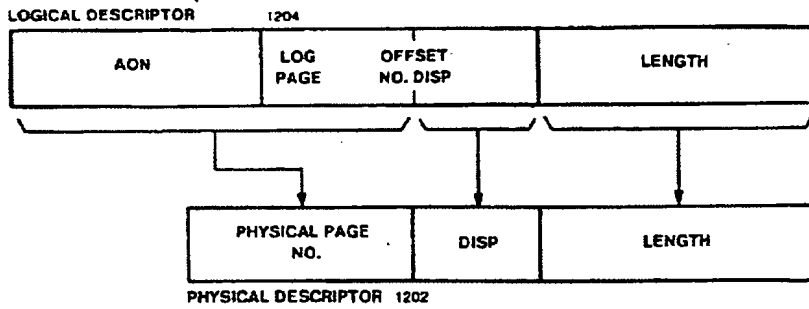


FIG 12

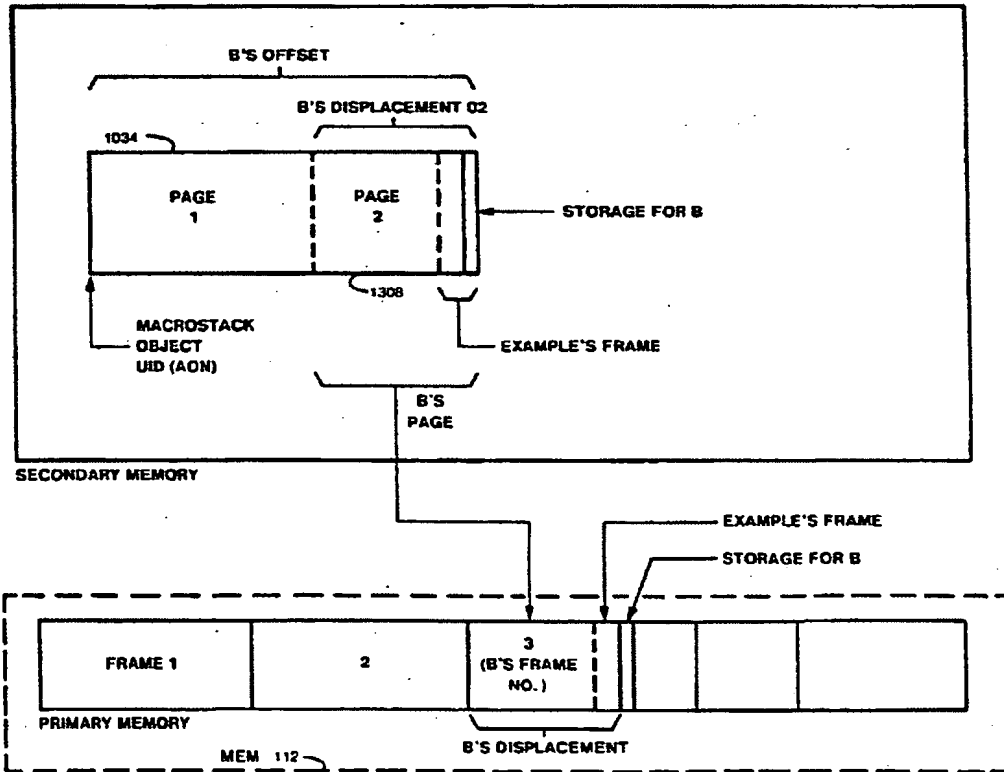


FIG 13

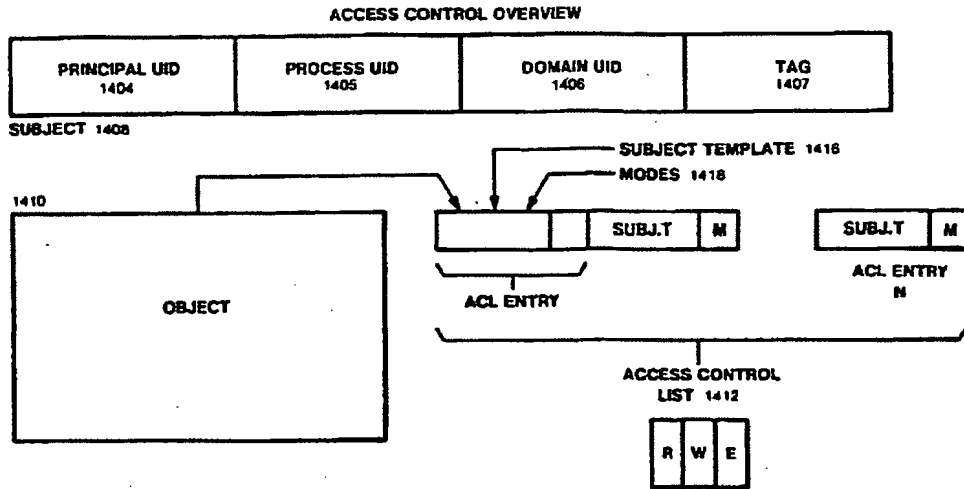


FIG 14

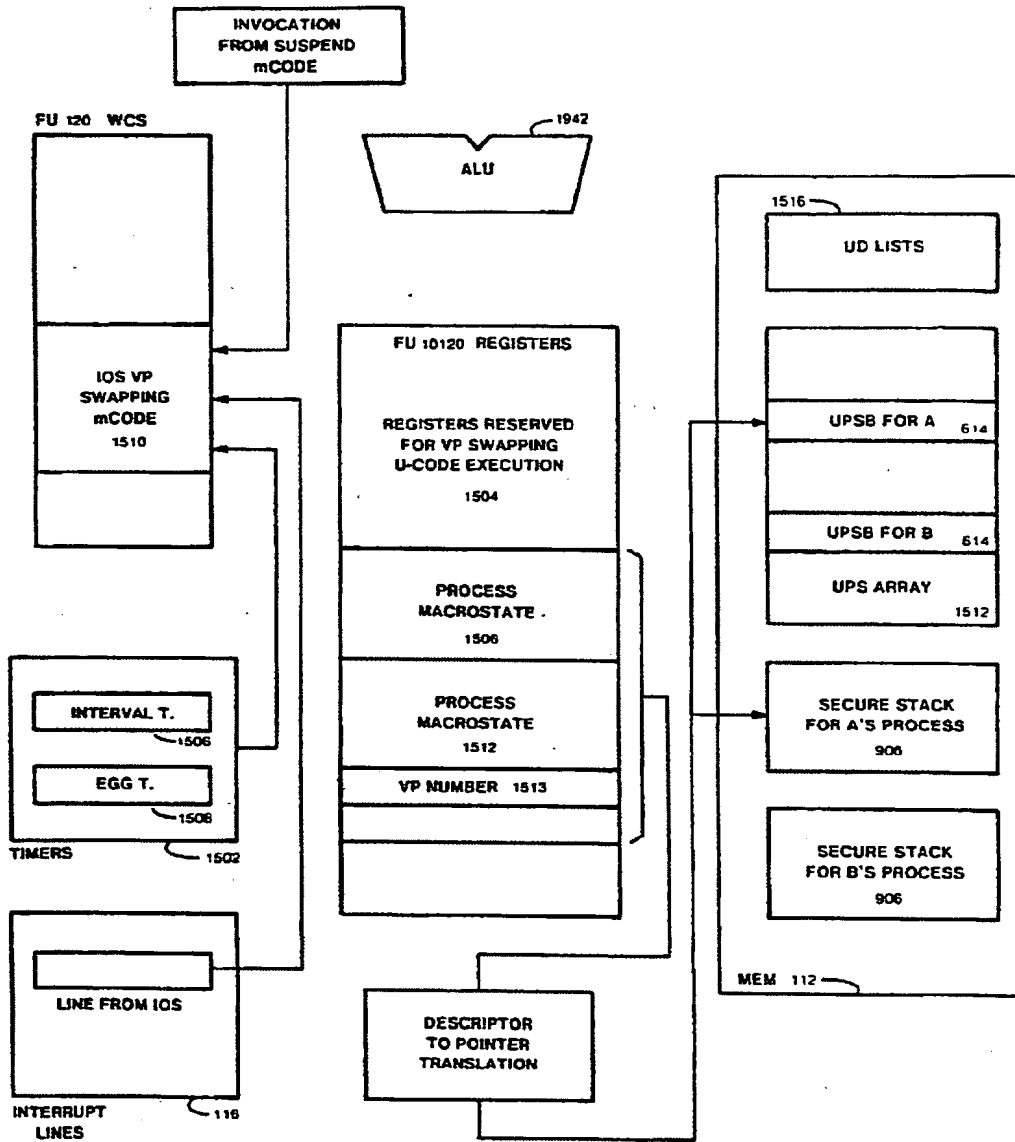


FIG 15

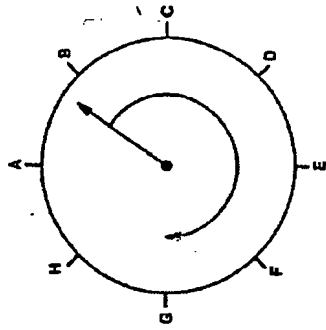


FIG 17

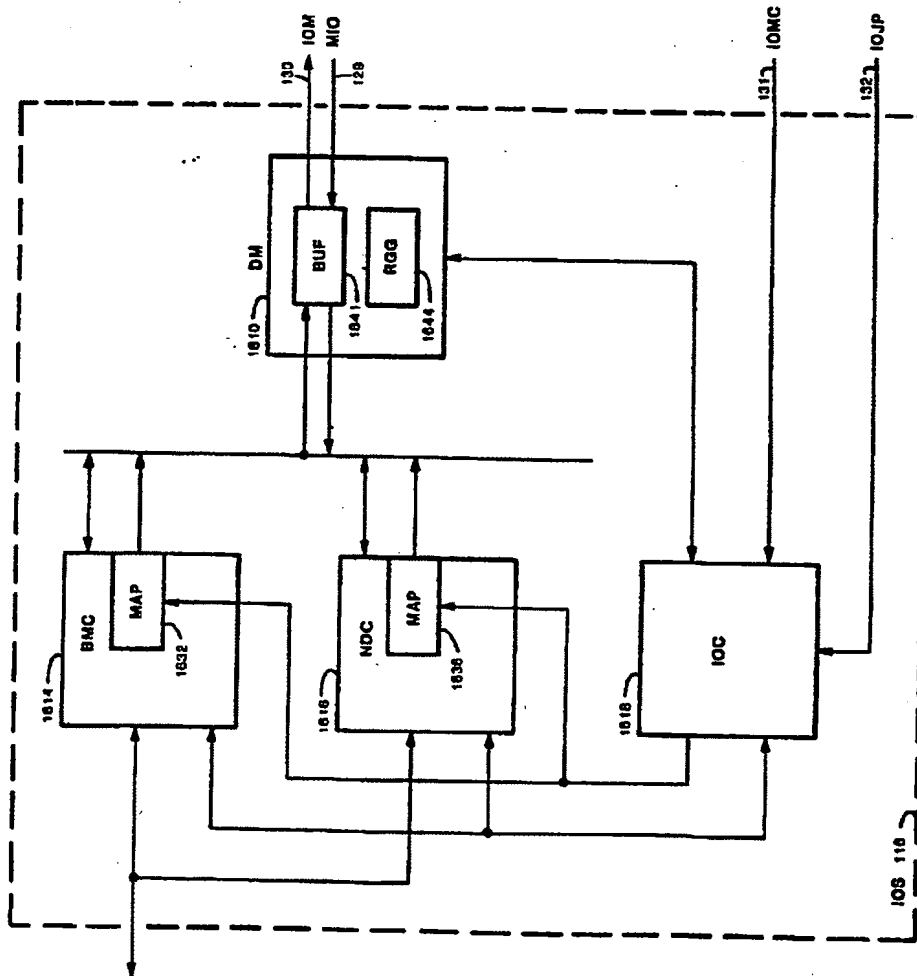


FIG 16

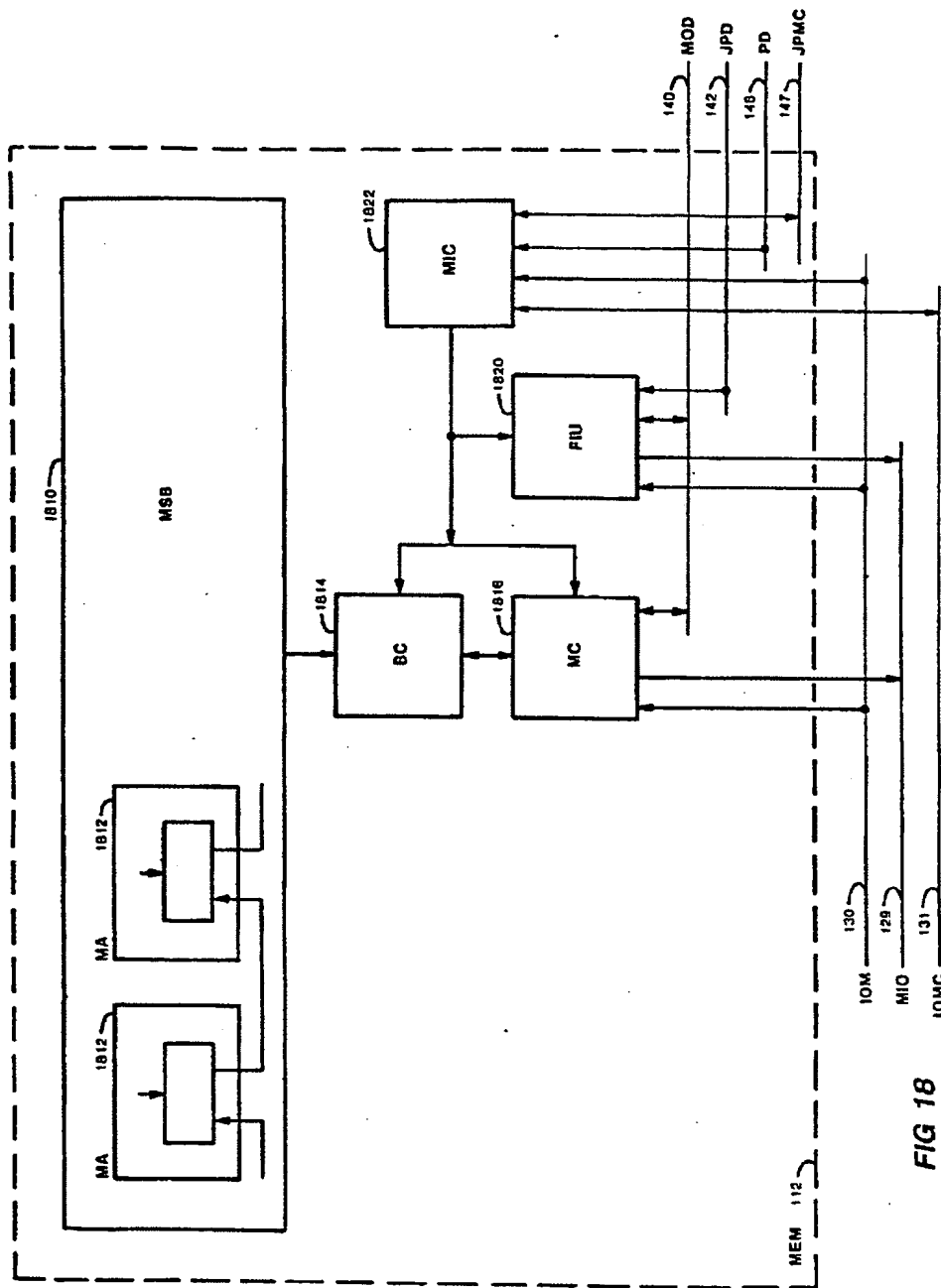


FIG 18

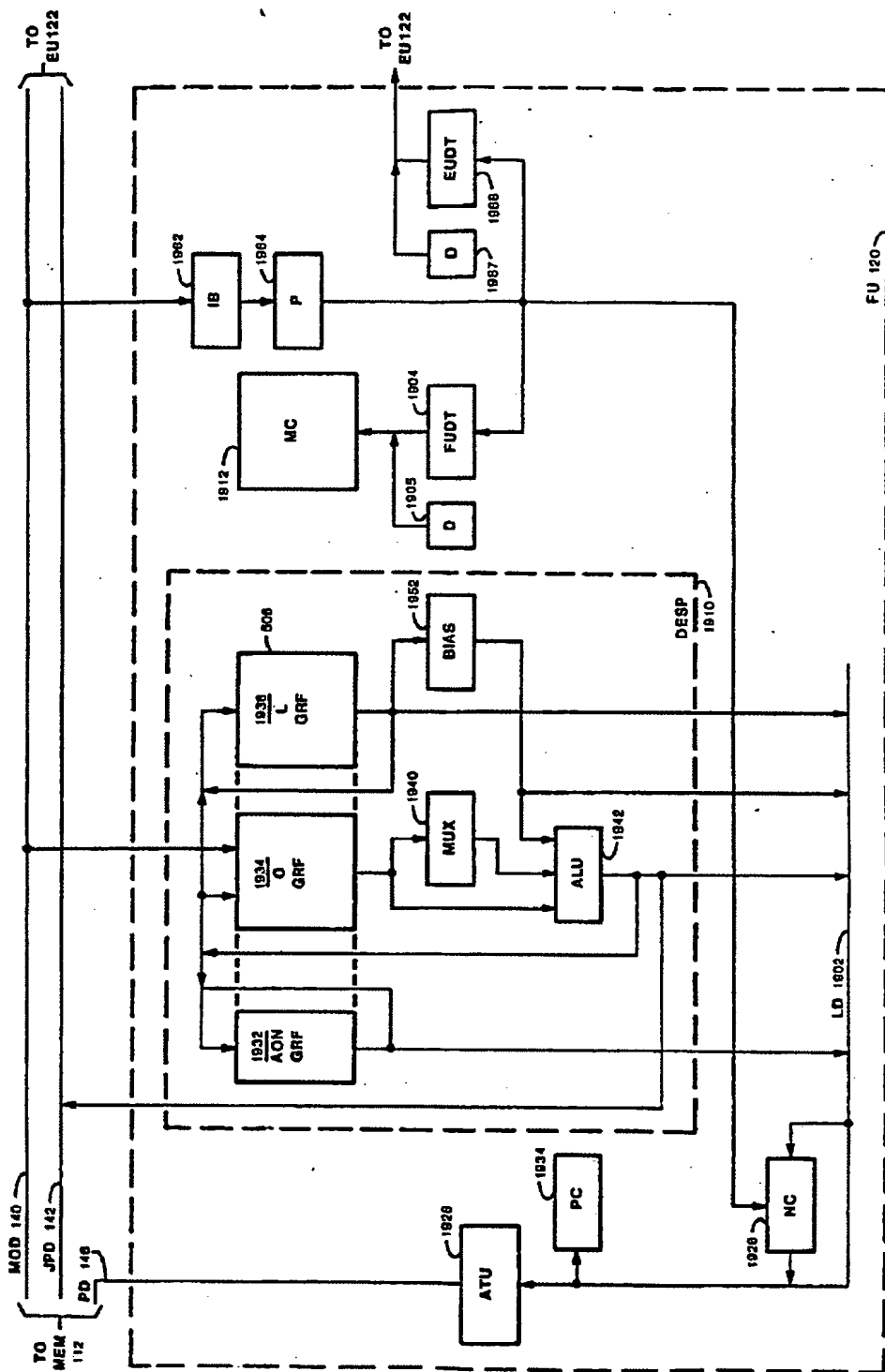


FIG 19

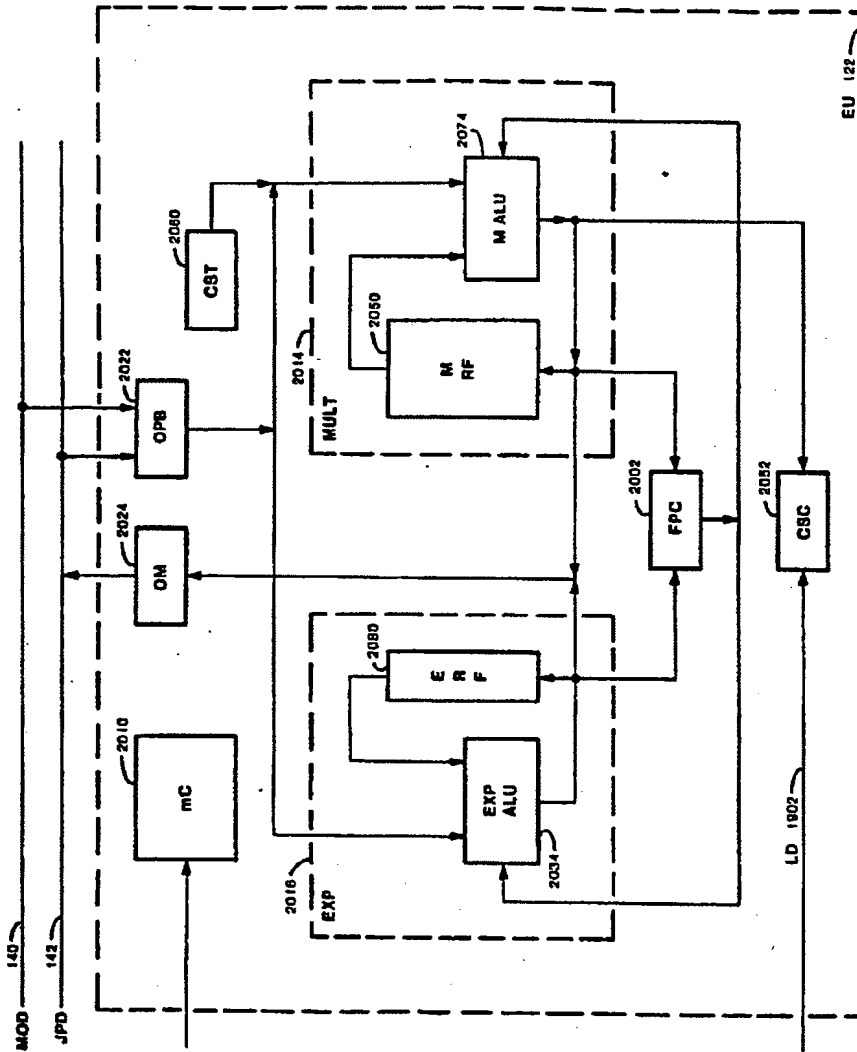


FIG 20

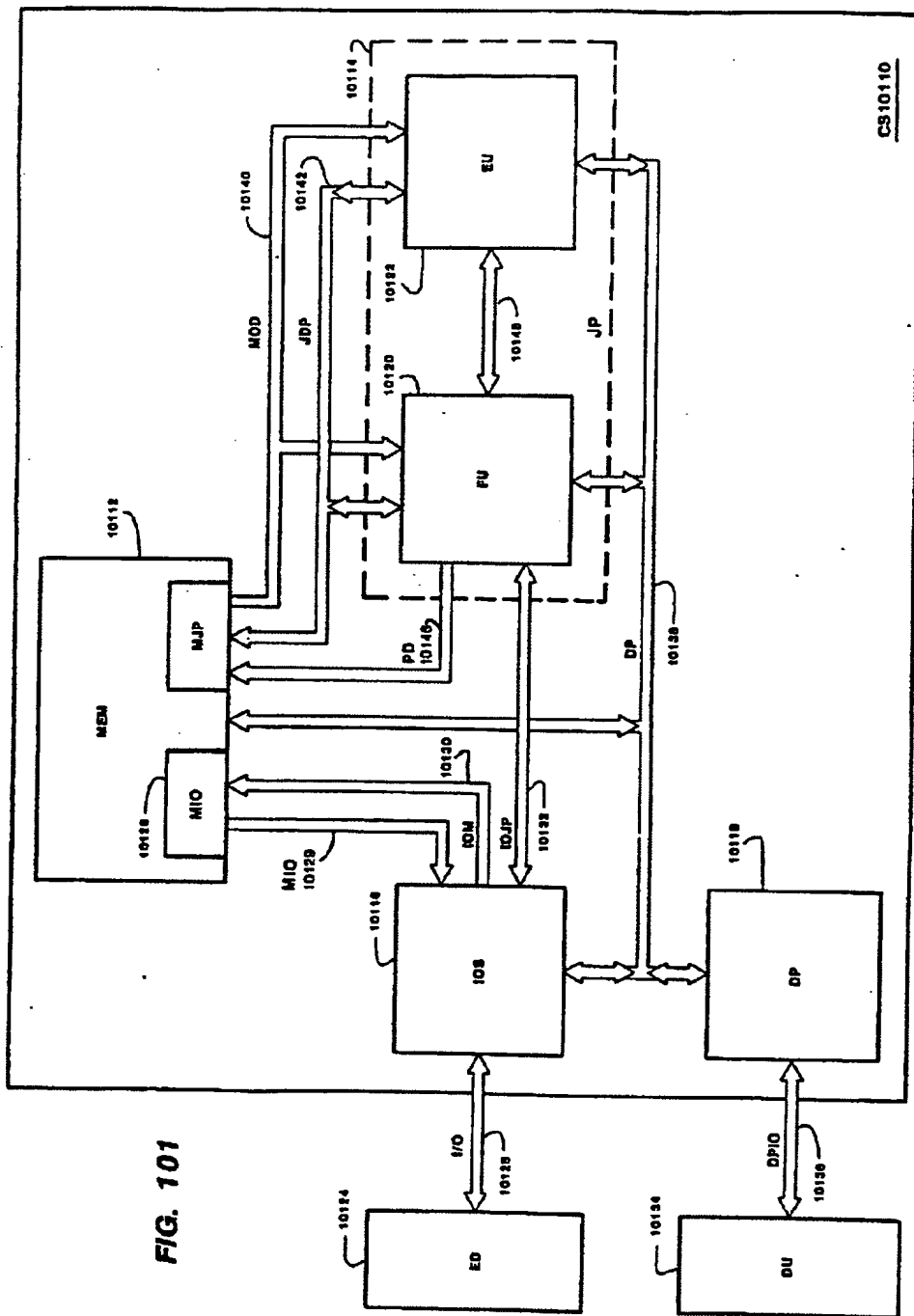


FIG. 101

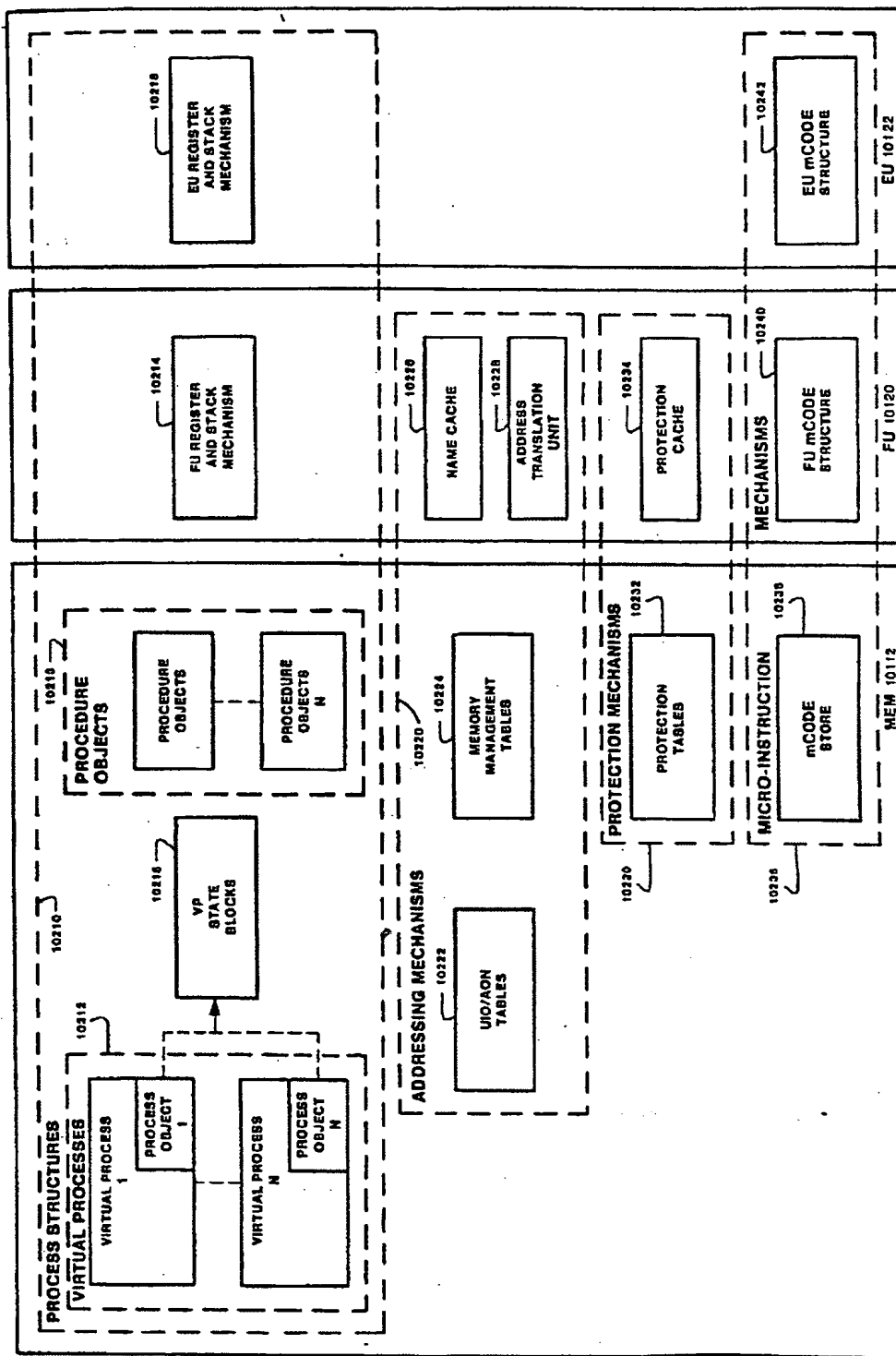


FIG. 102

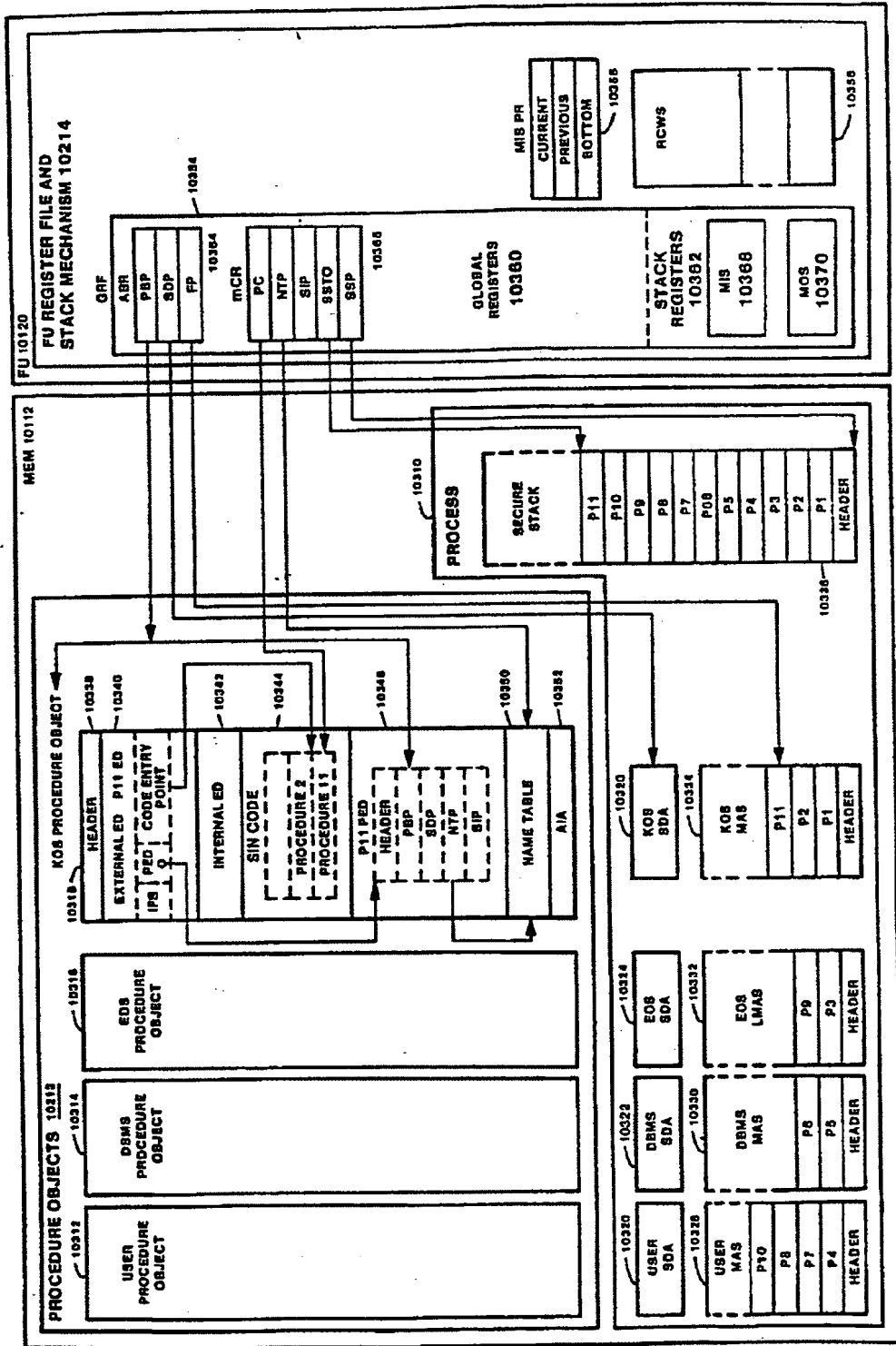


FIG. 103

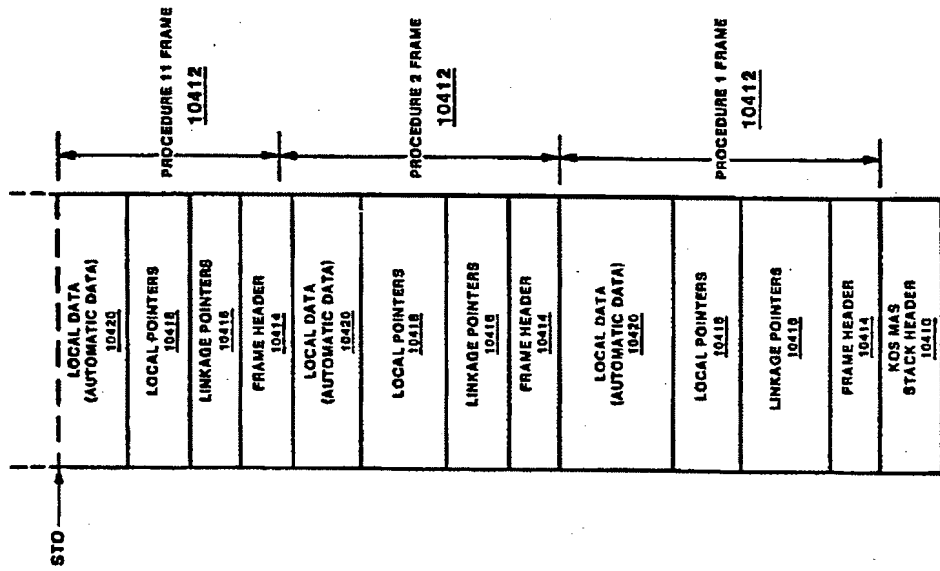


FIG. 104

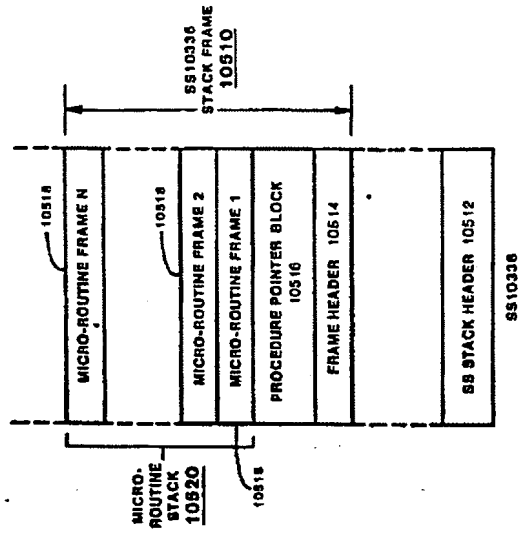


FIG. 105

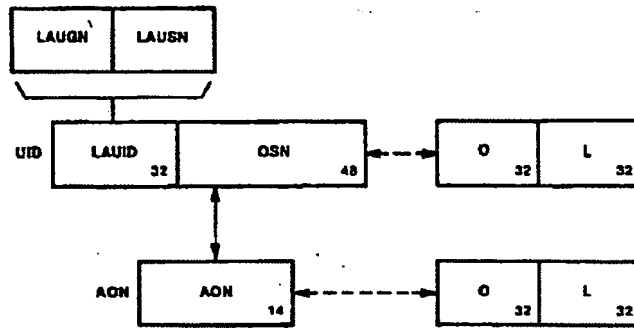


FIG. 106A

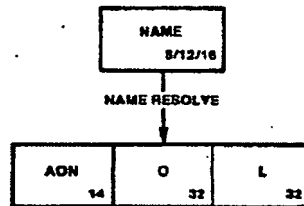


FIG. 106B

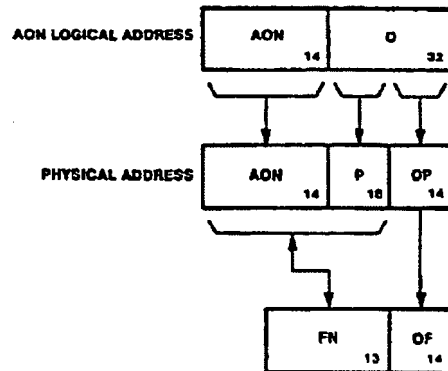


FIG. 106C

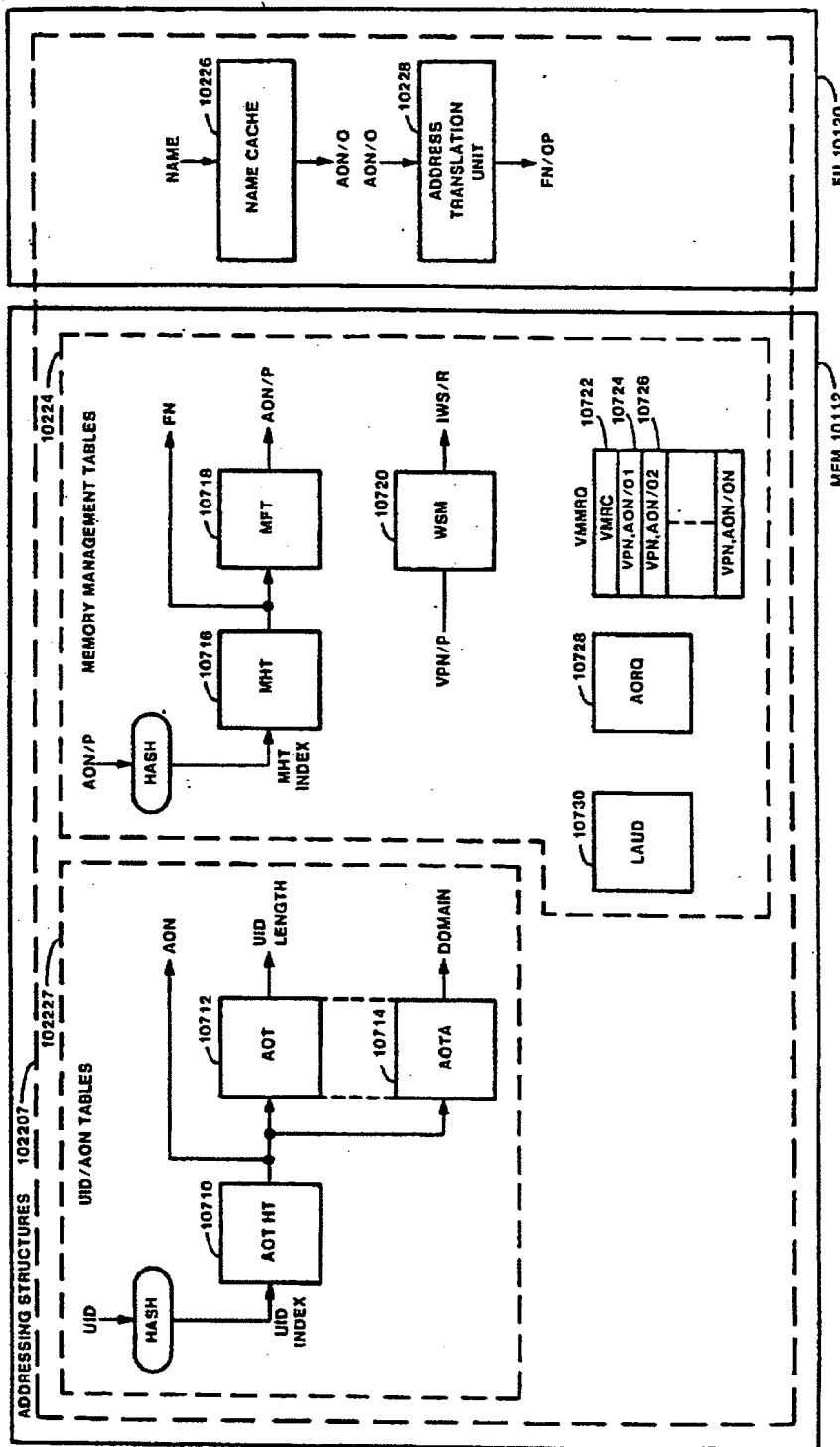


FIG 107

WORD A NTE

FLAG	B	PR
L	D	

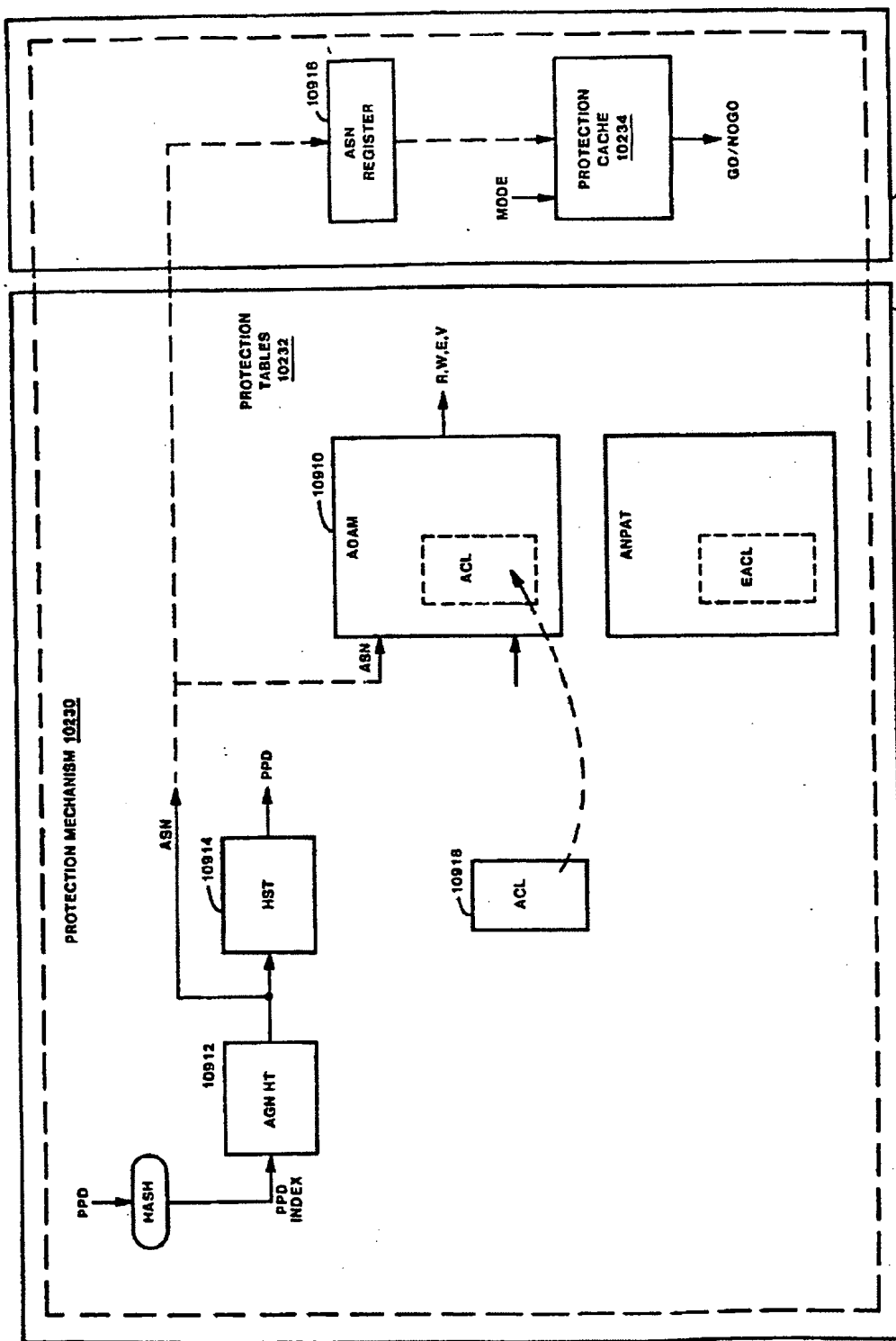
WORD B

WORD C

D	I
IES	

WORD D

FIG. 108



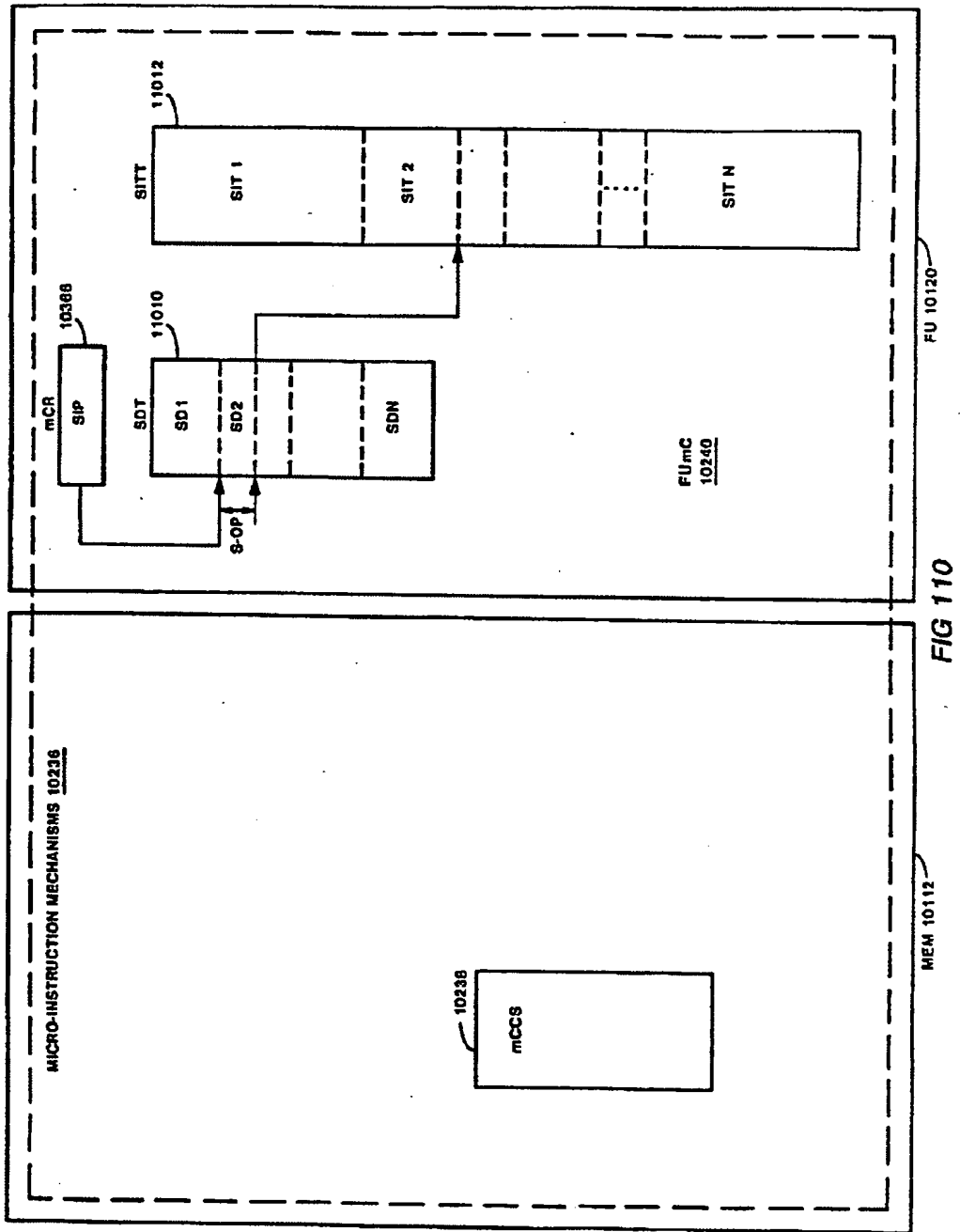


FIG 110

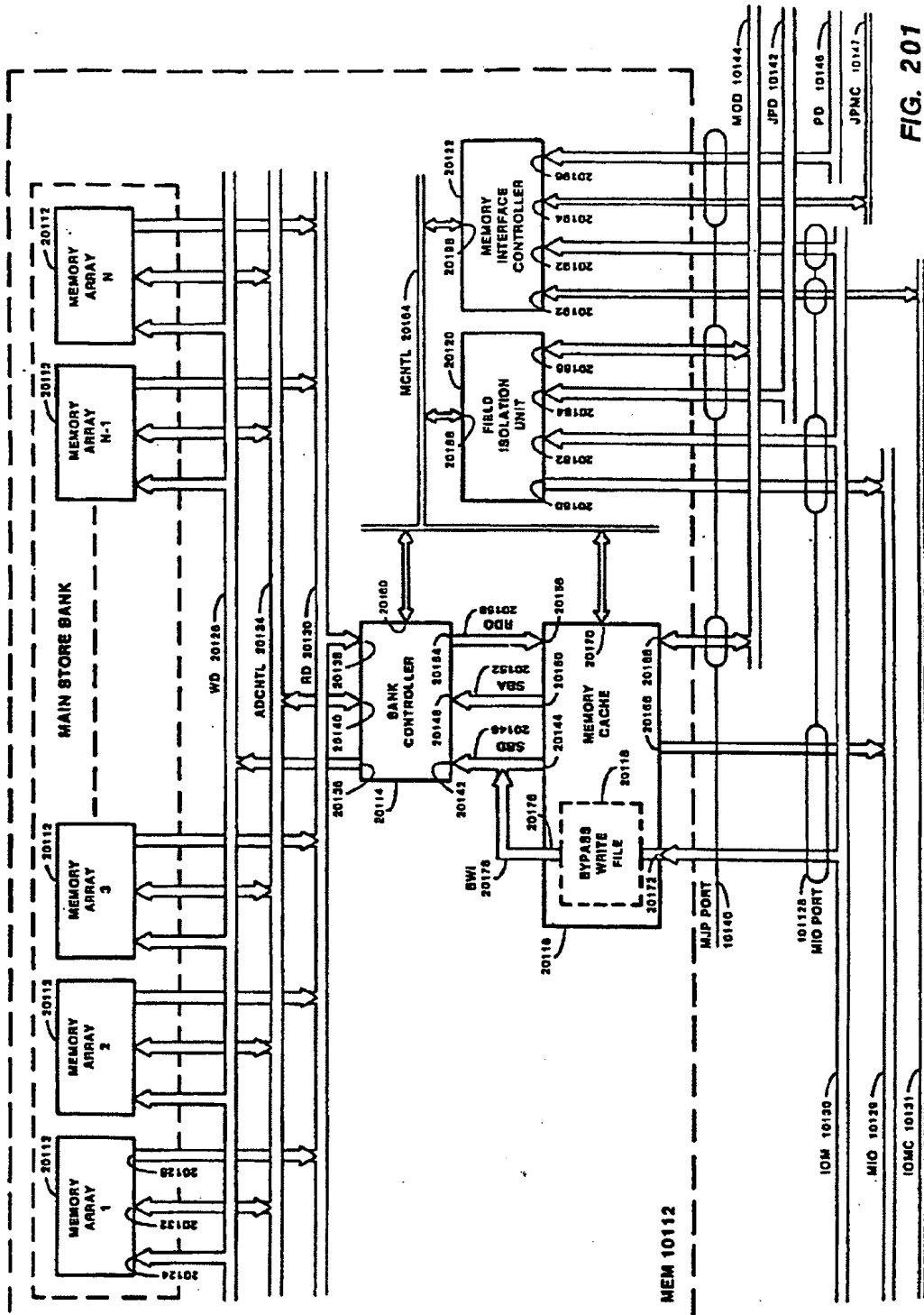


FIG. 201

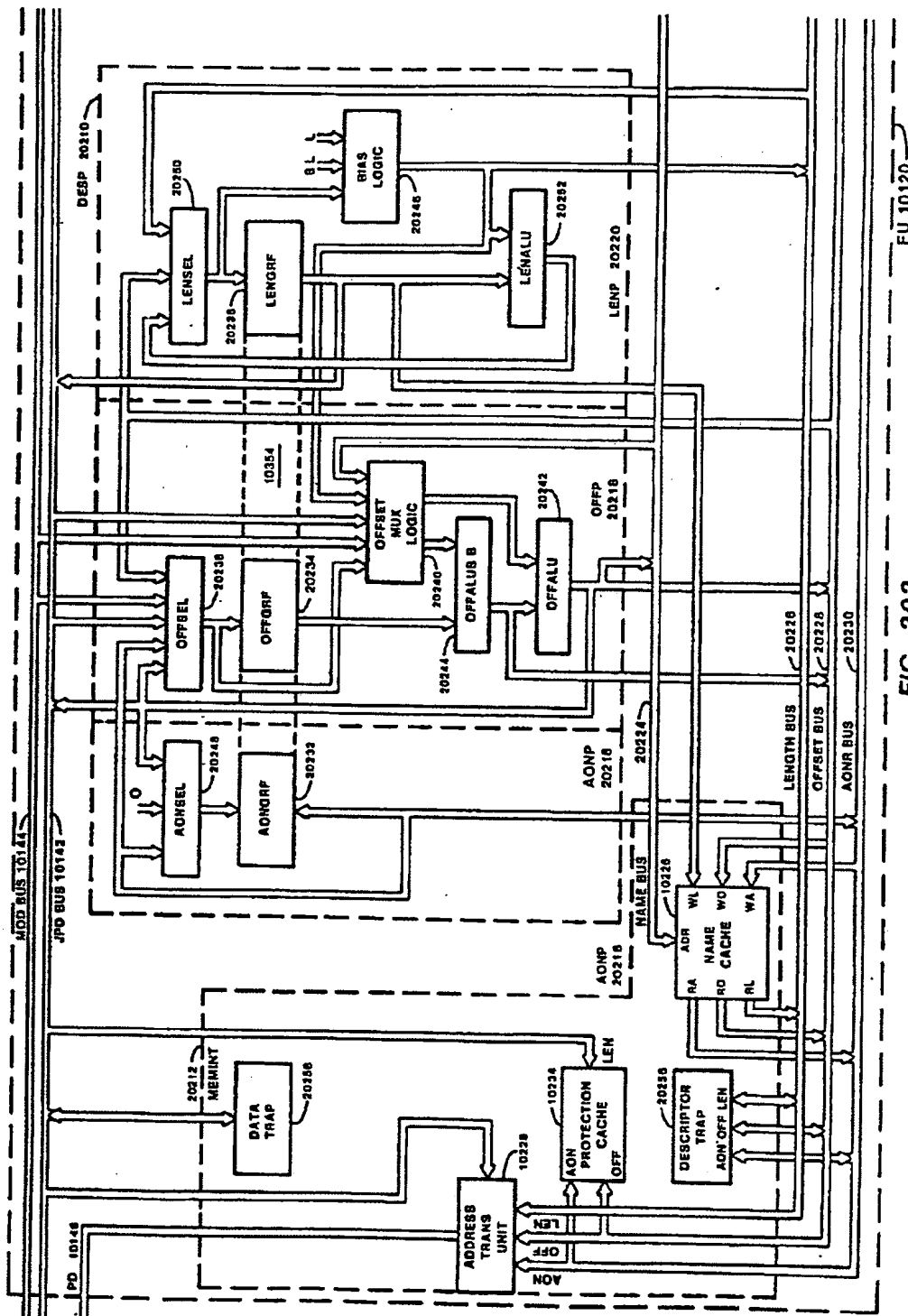


FIG. 202

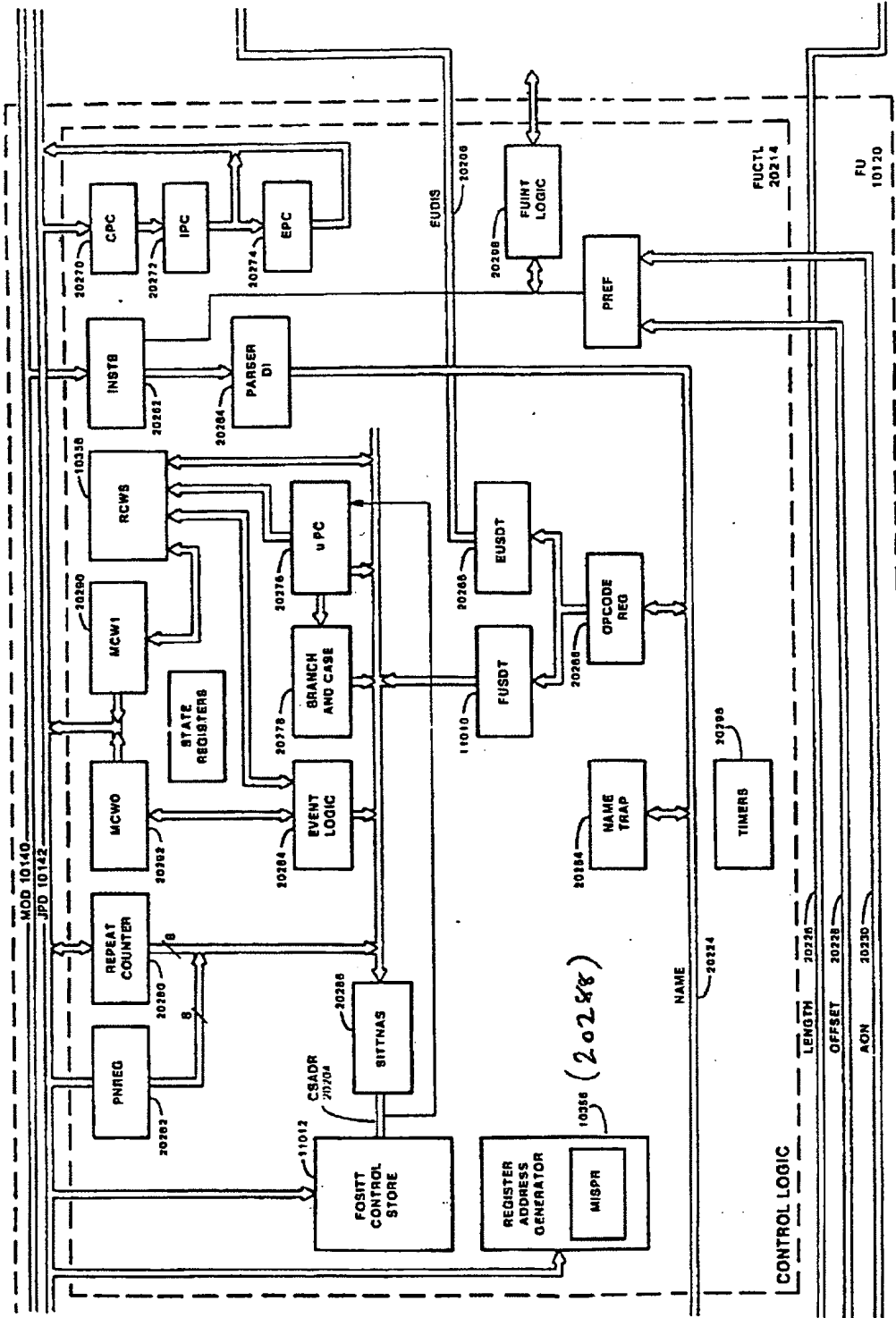


FIG. 202A

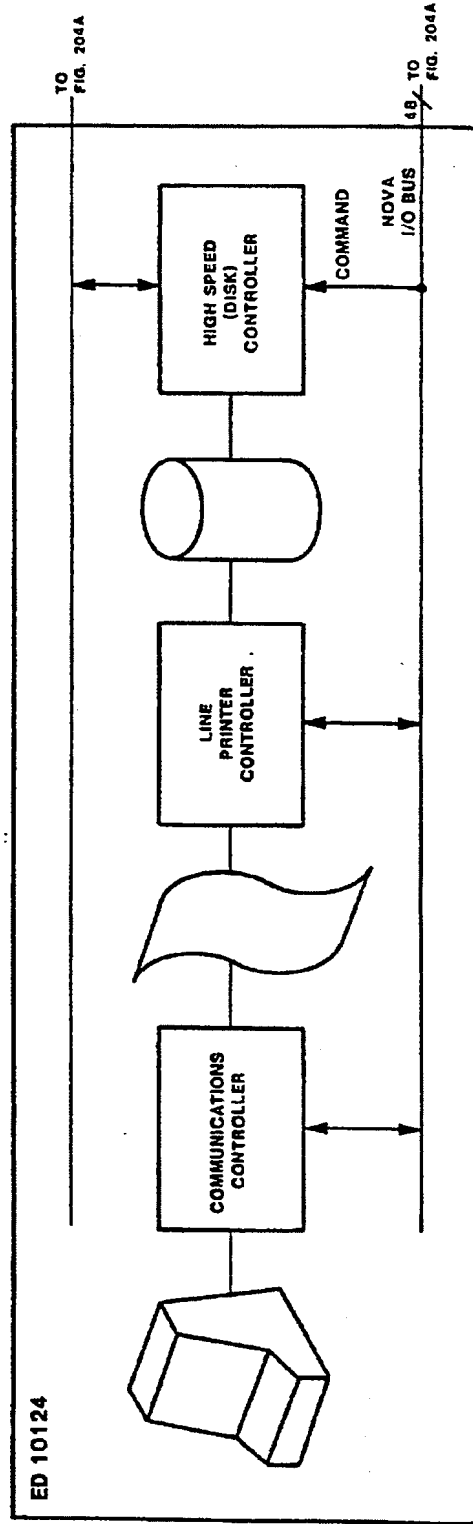


FIG. 204

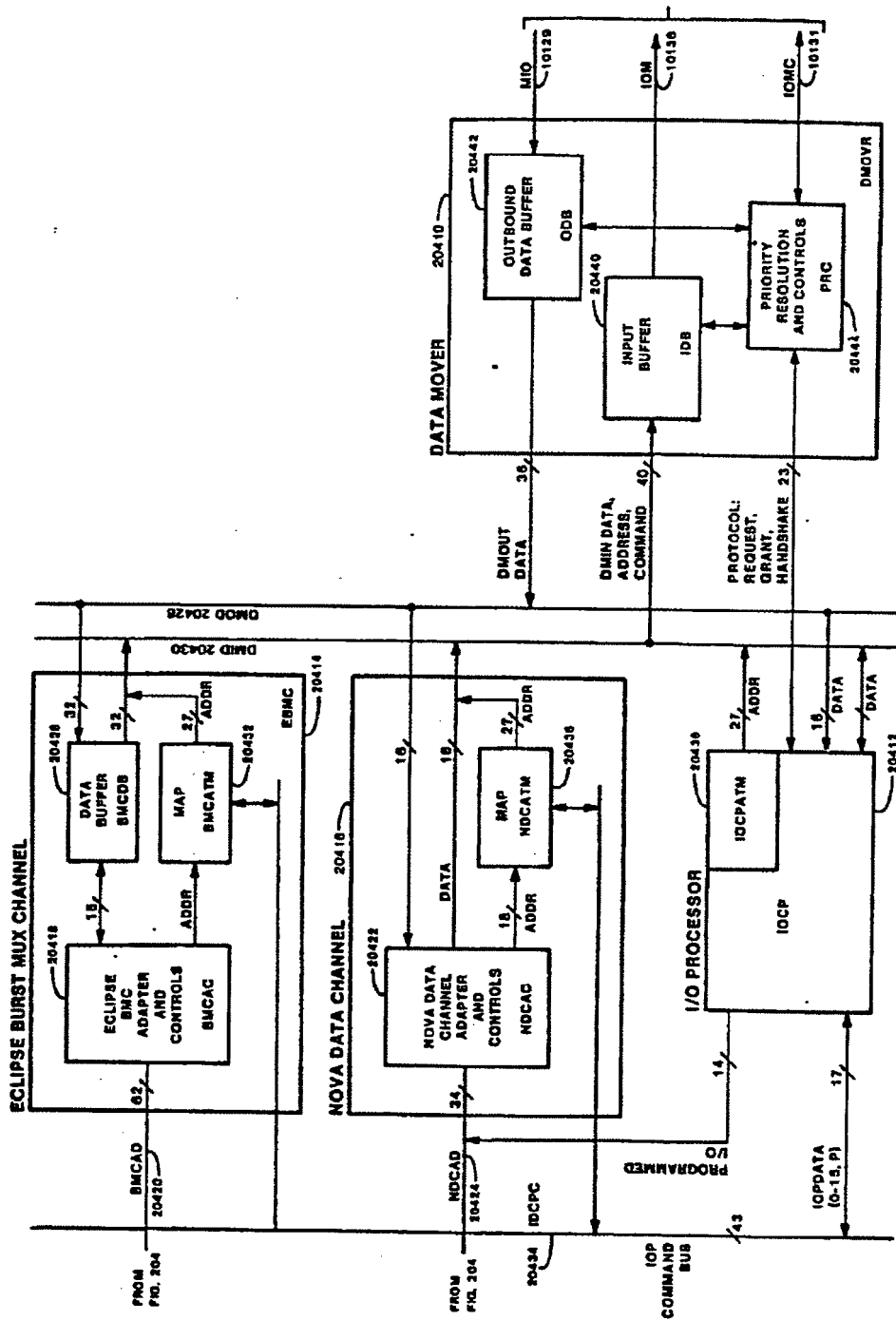


FIG. 204A

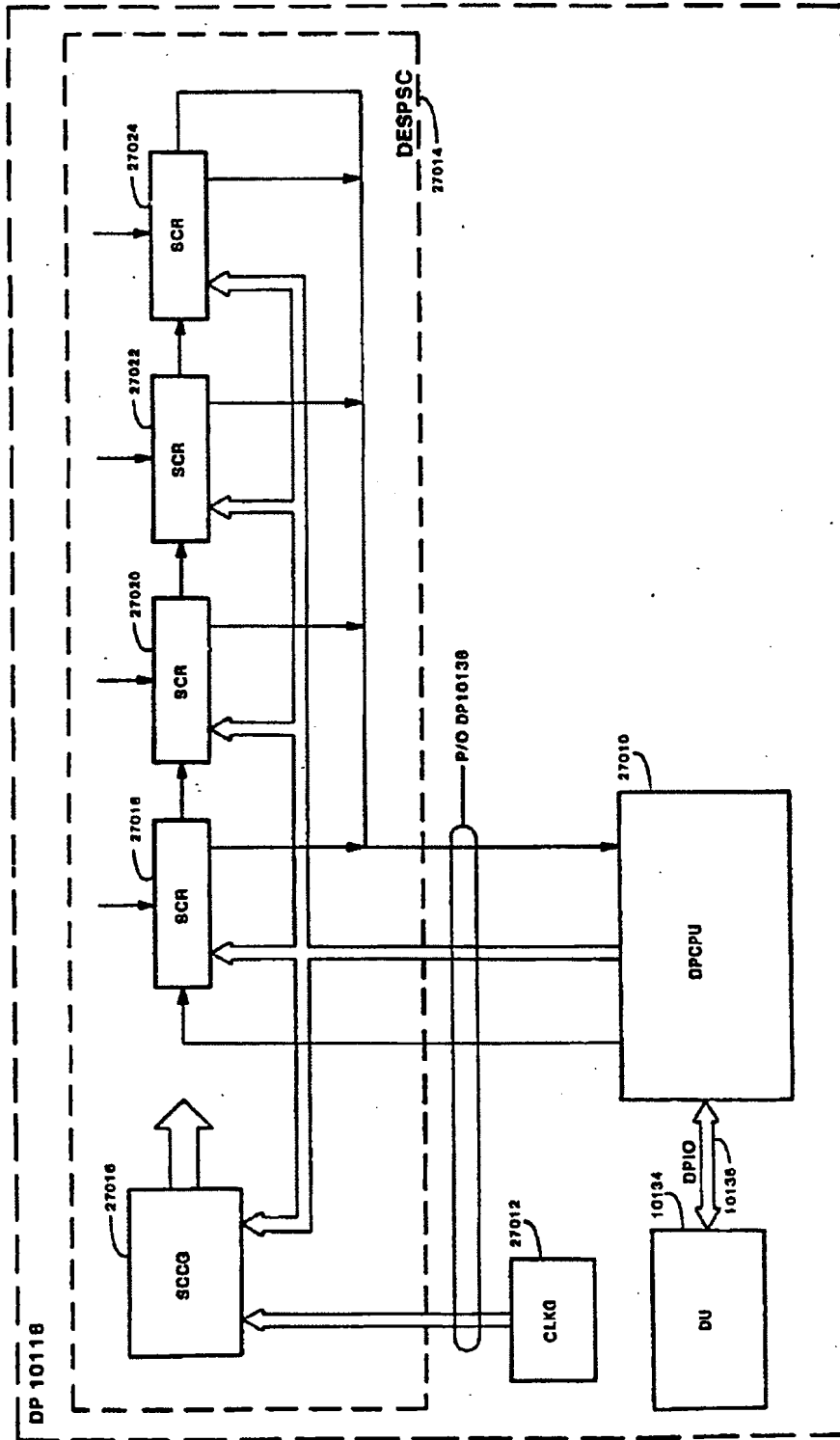


FIG. 205

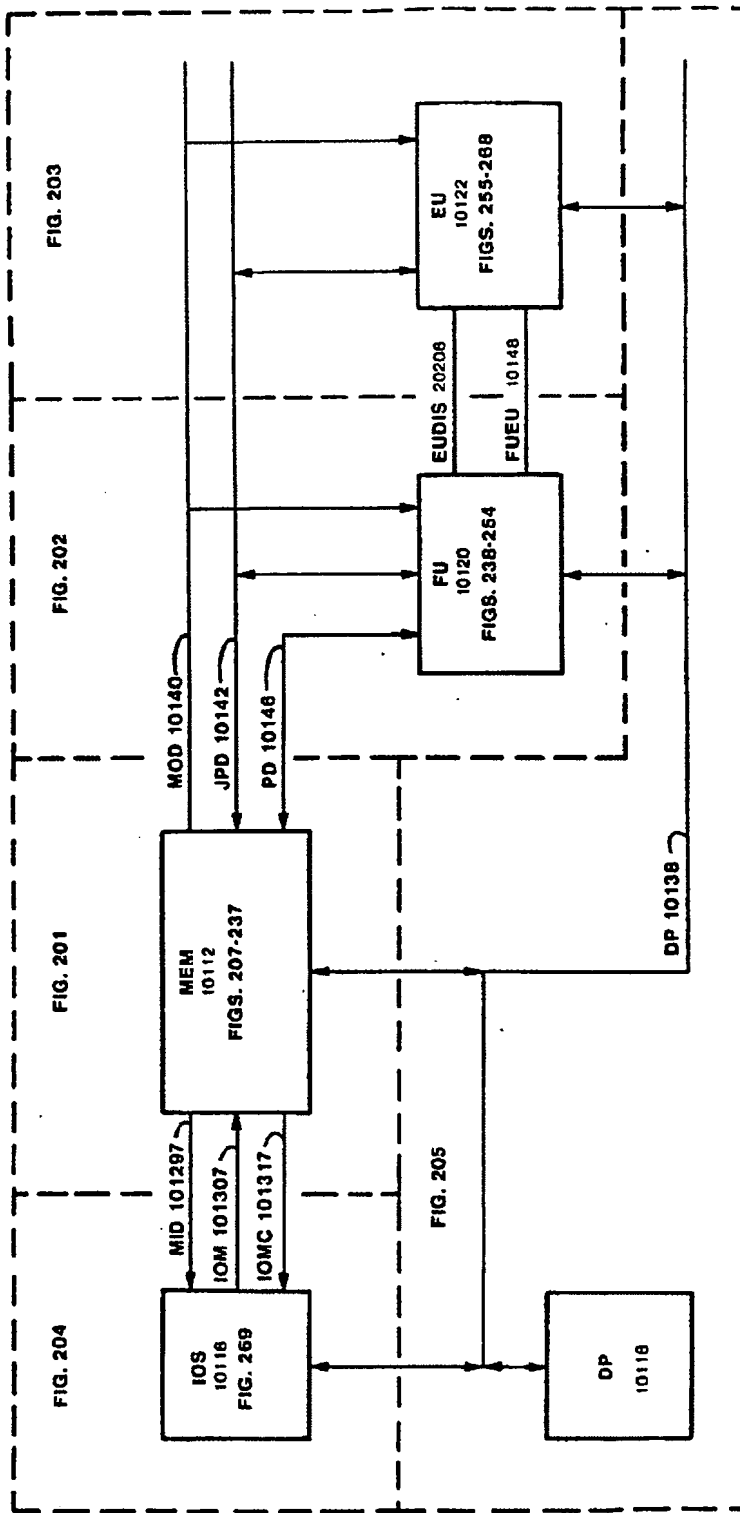


FIG. 206

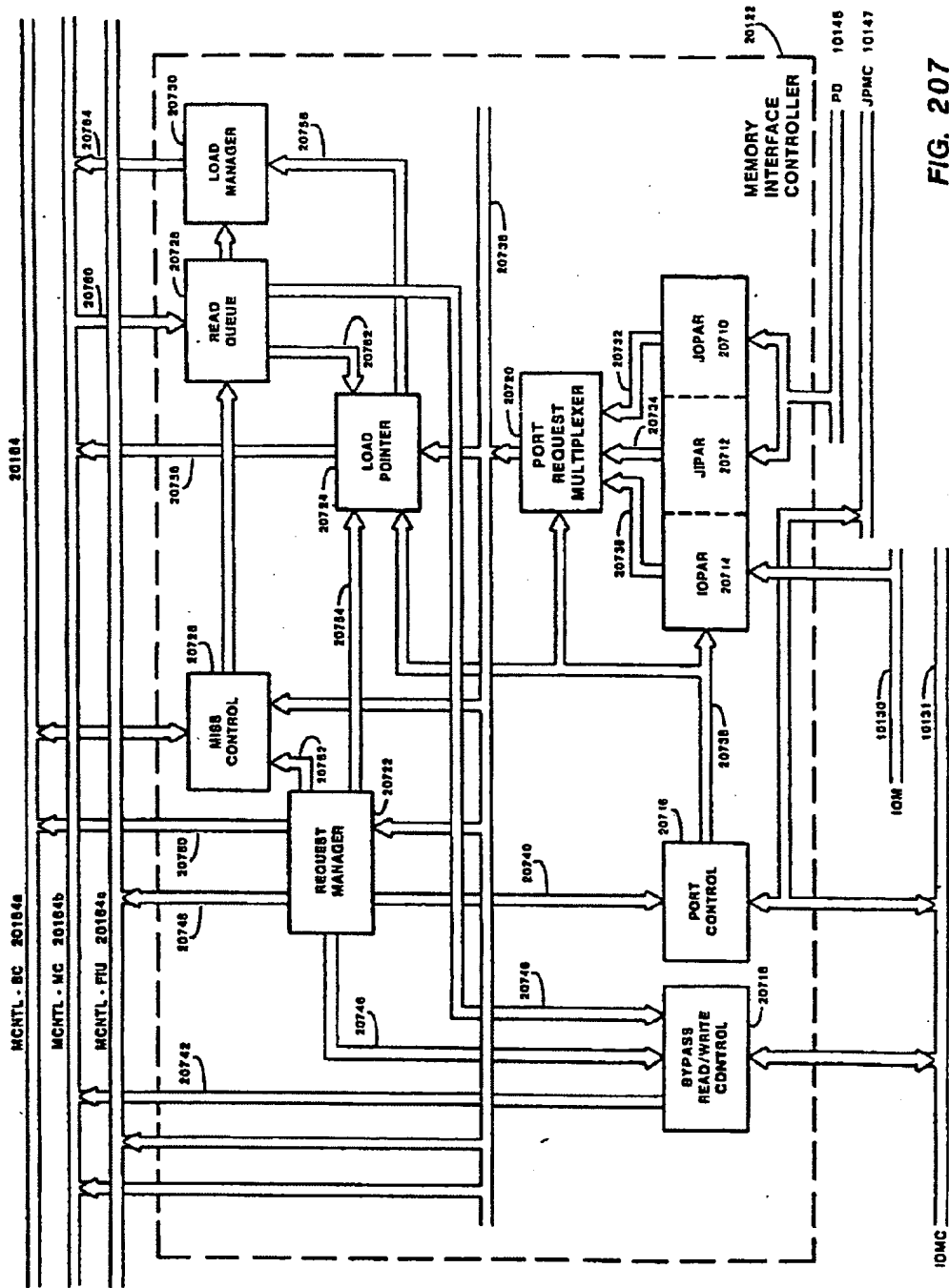
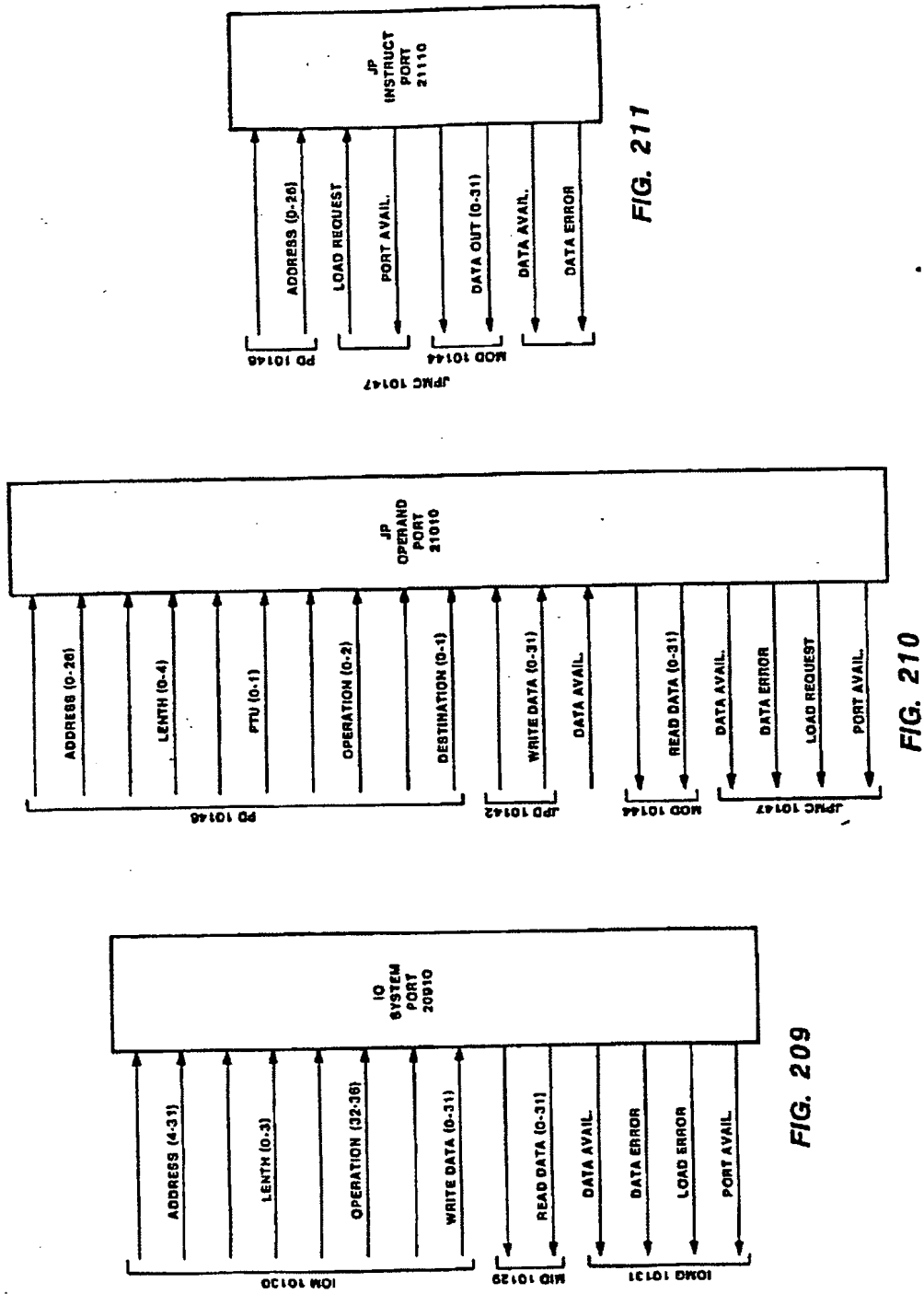


FIG. 207



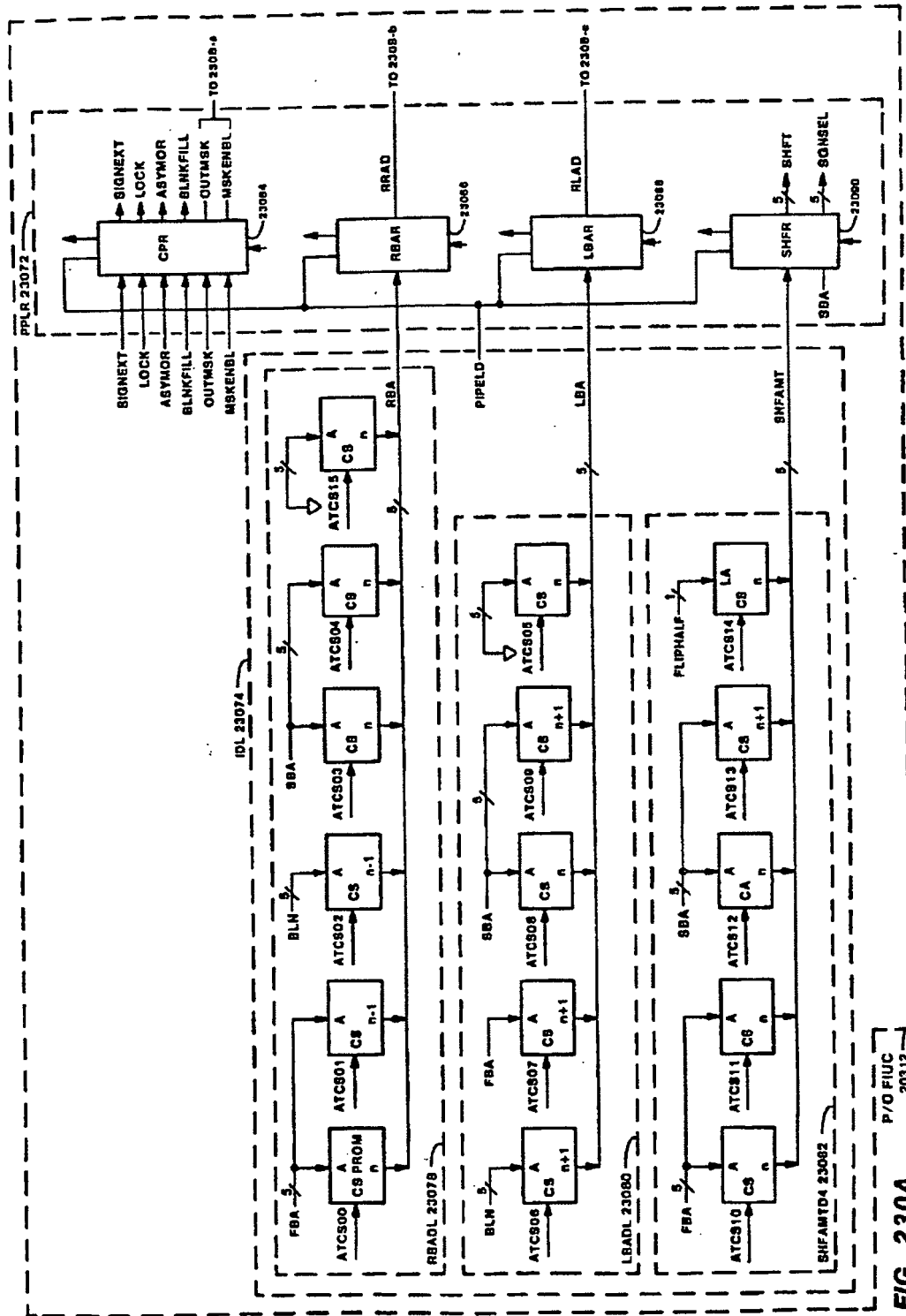
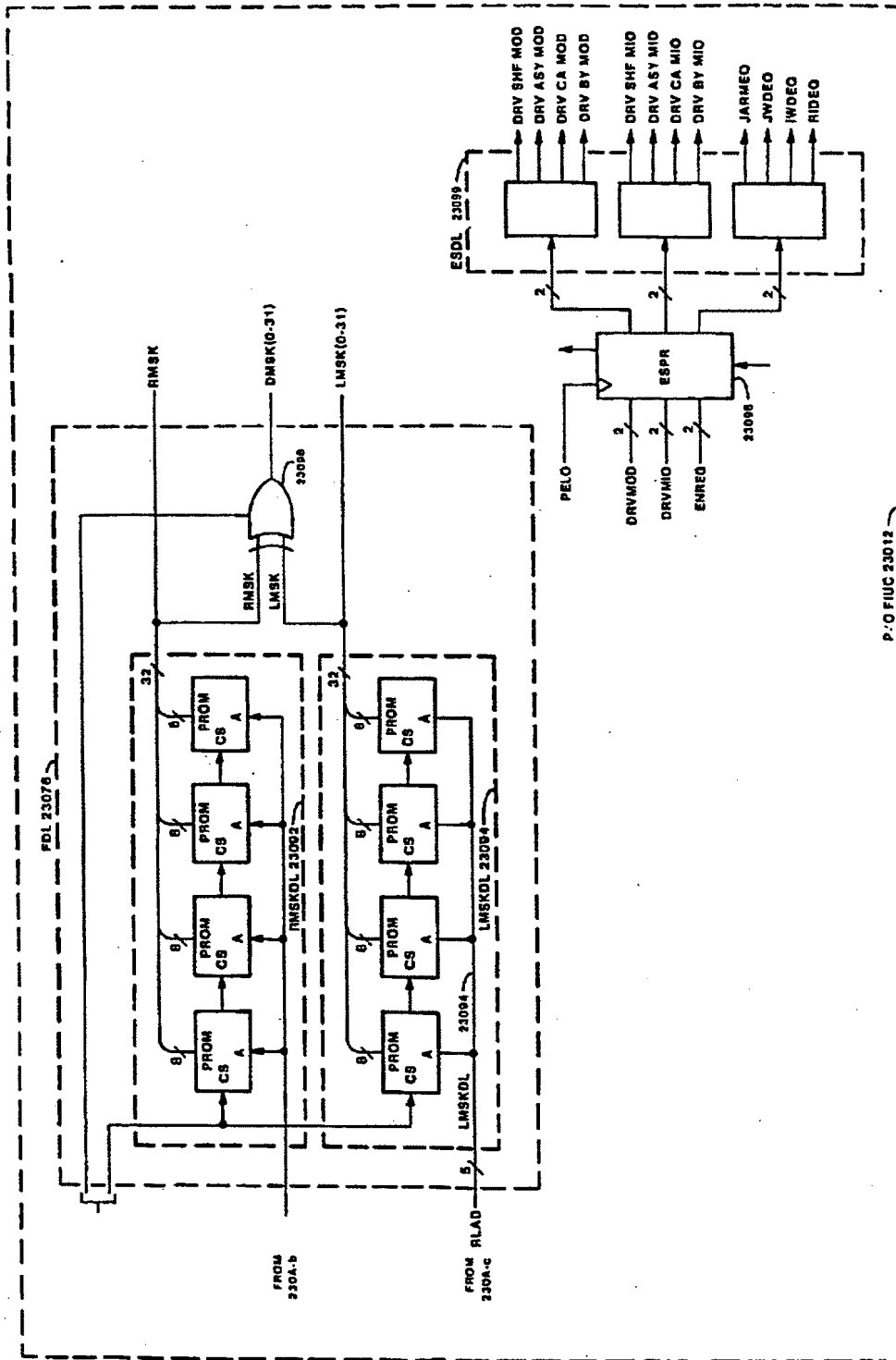


FIG. 230A



P.O. FIUC 23012

FIG 230B

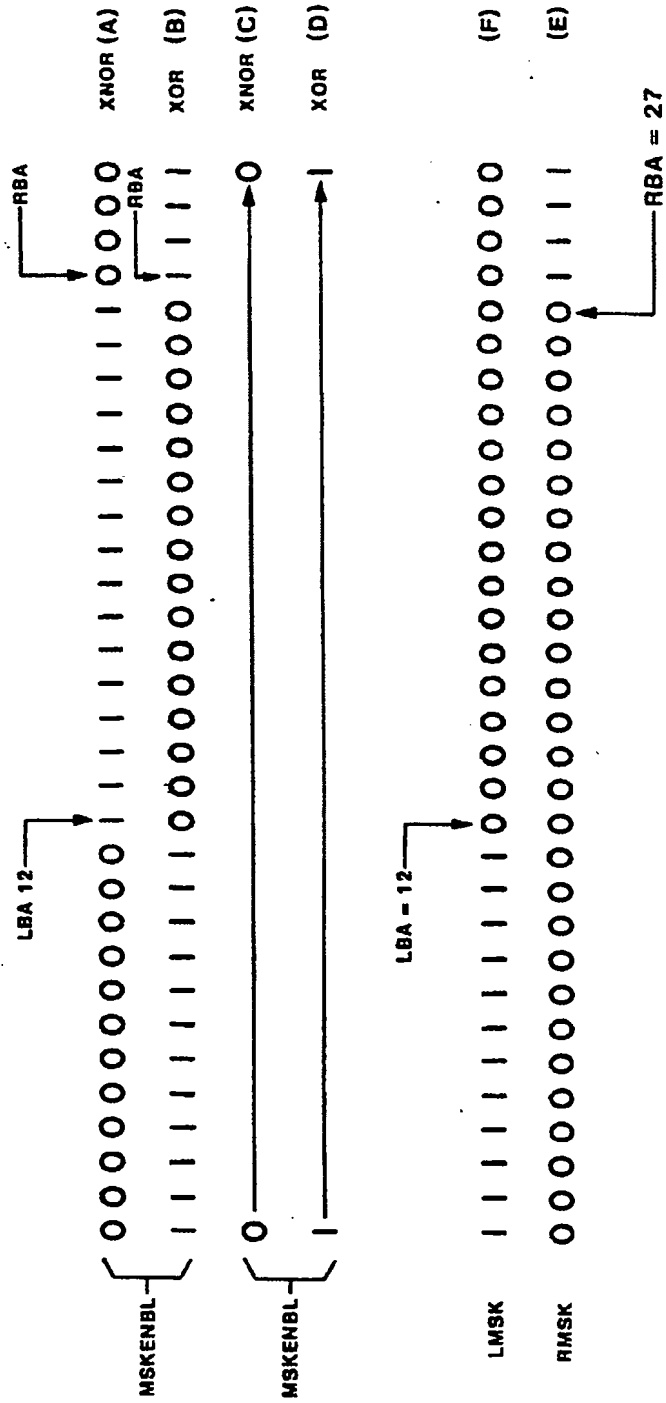
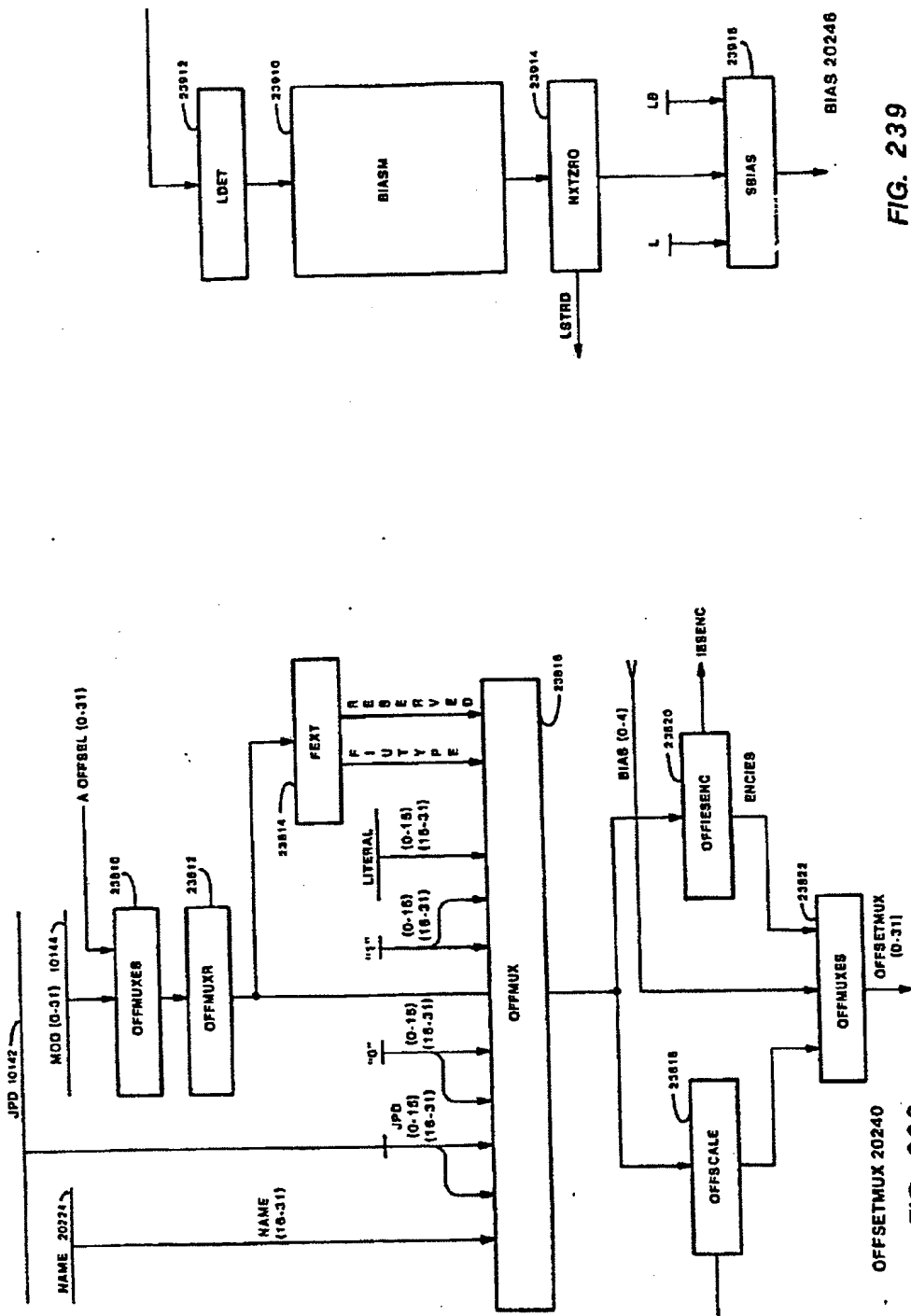
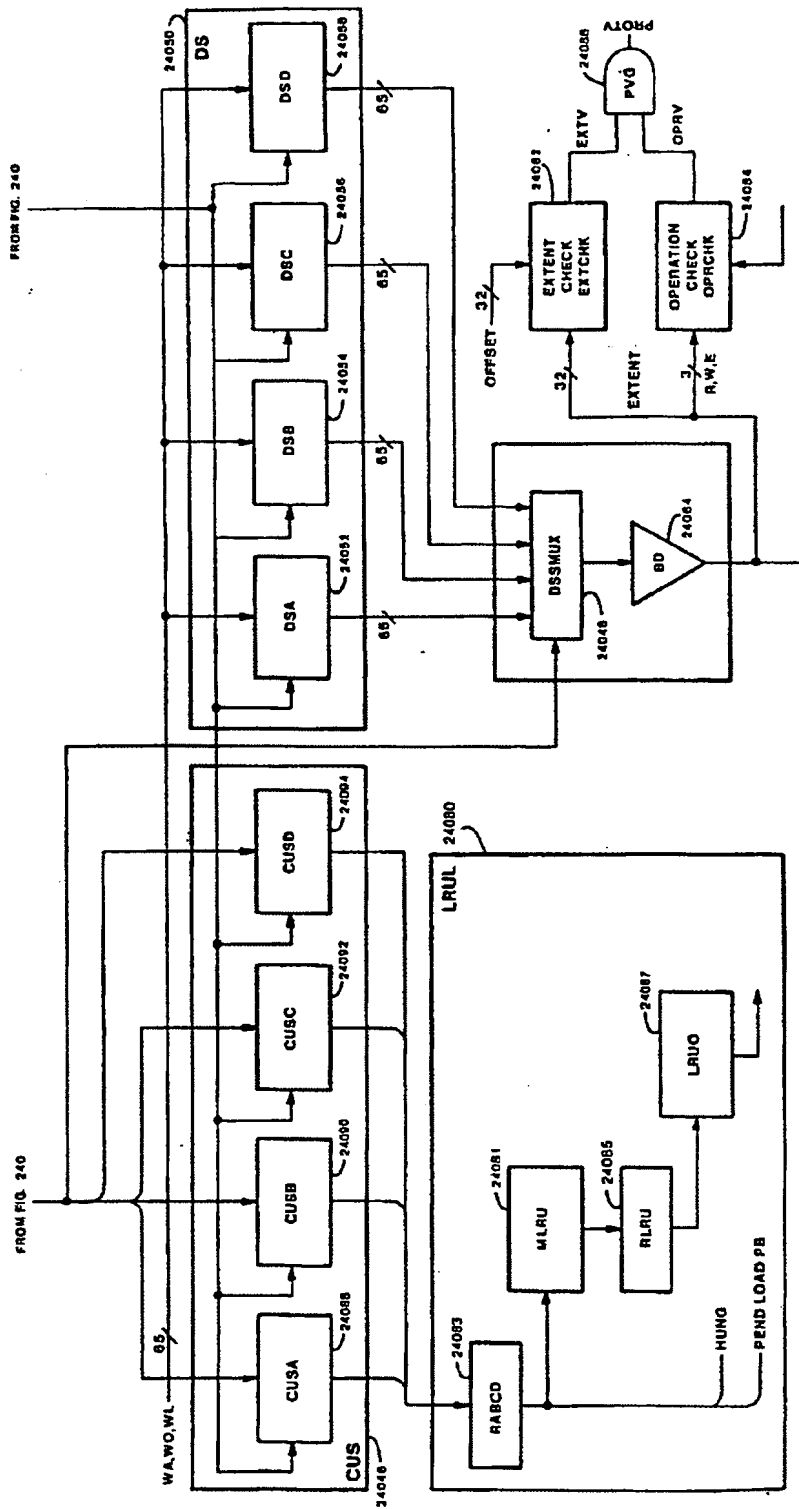


FIG. 231





NC 10226

FIG. 240A

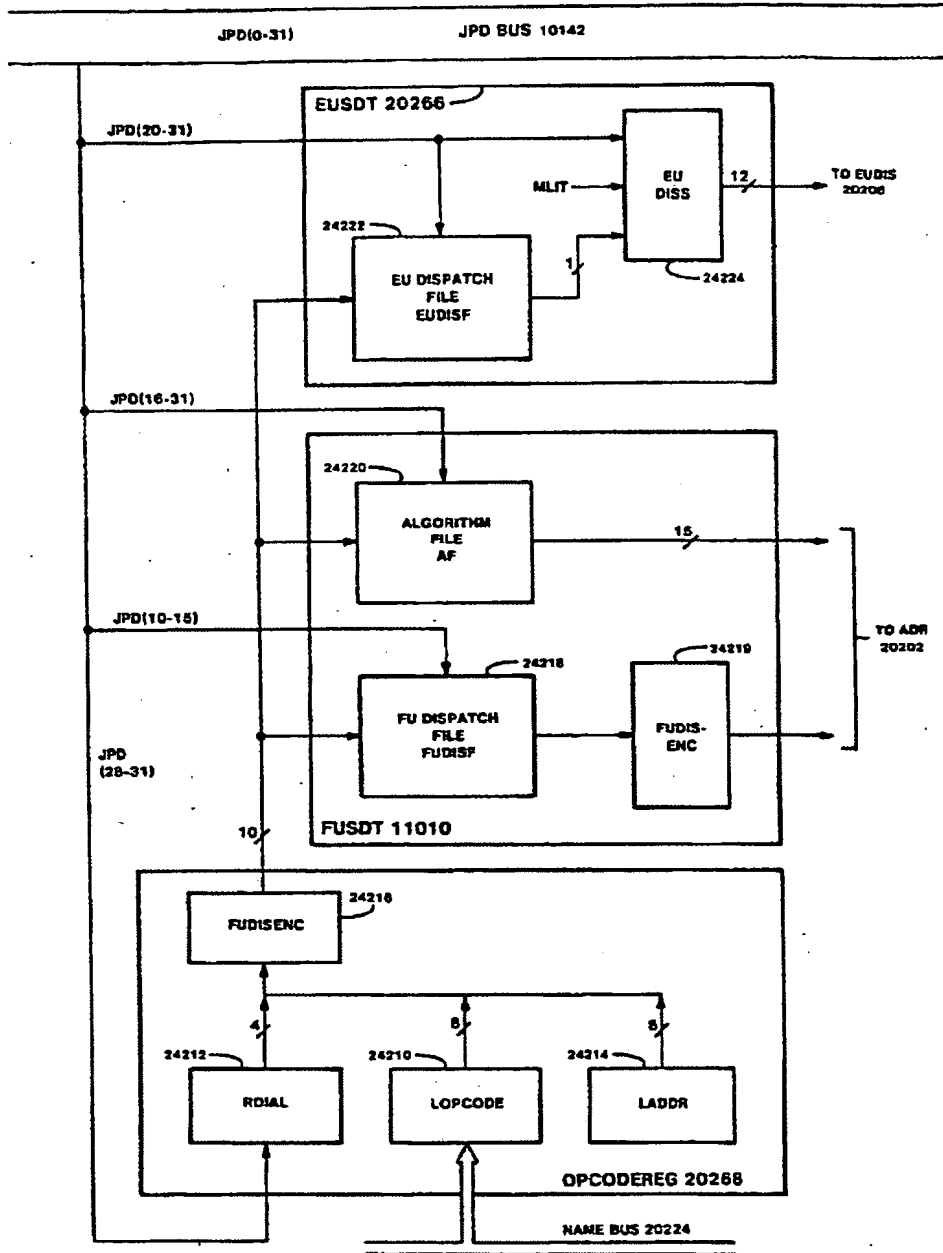


FIG. 242

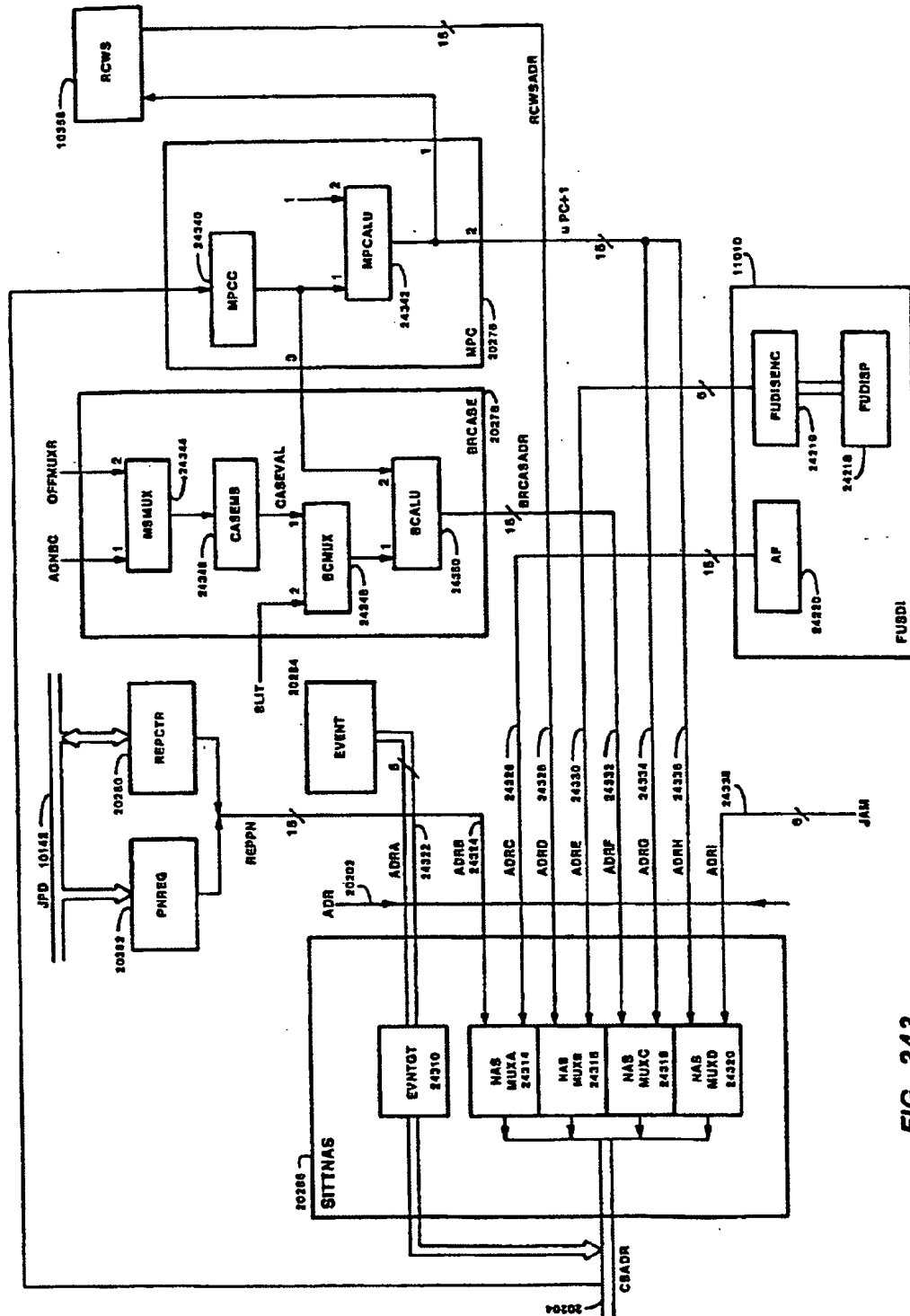


FIG. 243

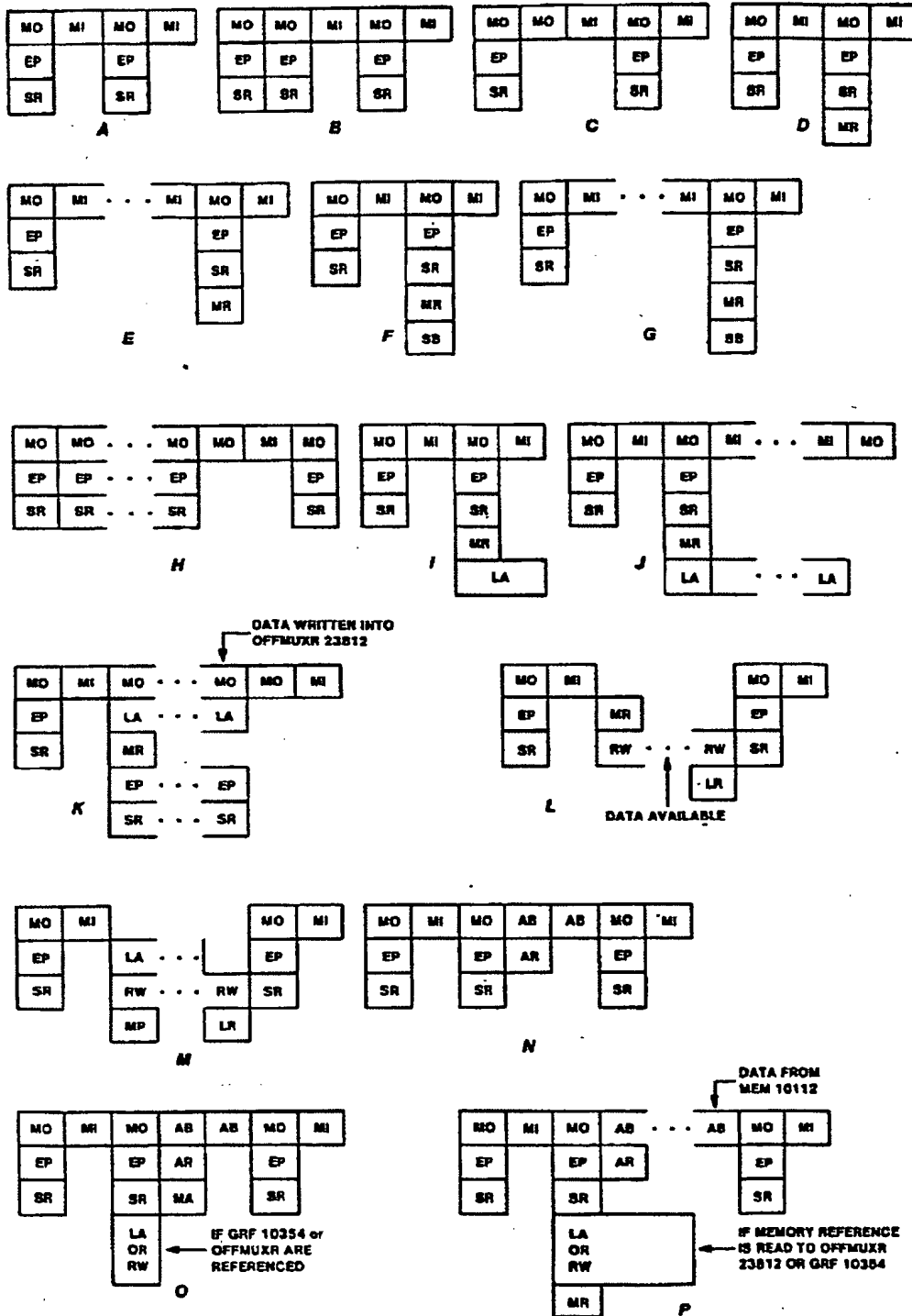


FIG. 244

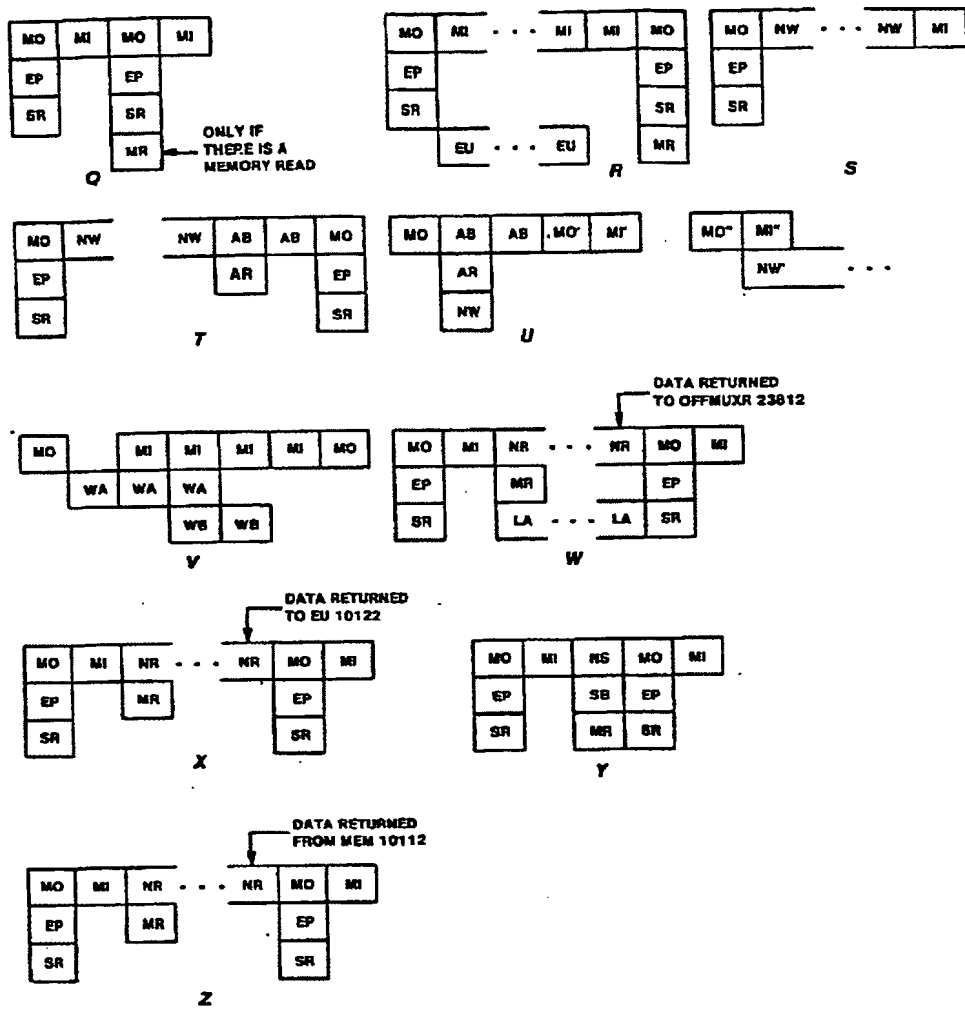


FIG. 244A

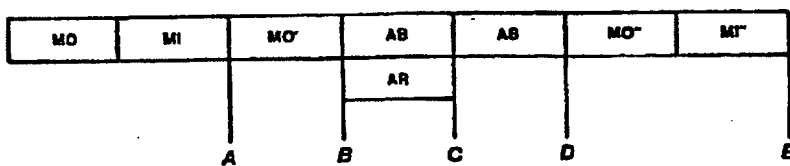


FIG. 245

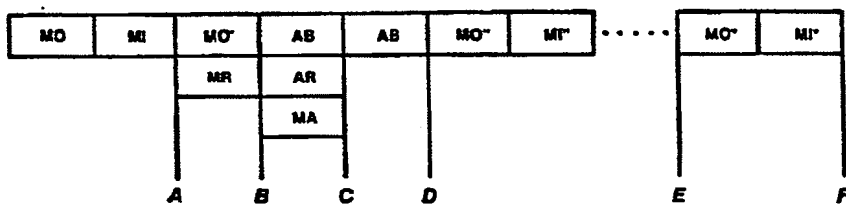


FIG. 246

PRIORITY LEVEL	EVENT	MASKED BY
0	E-UNIT STACK OVERFLOW	NONE
1	FATAL MEMORY ERROR	NONE
2	POWER FAIL	I
3	F-BOX STACK OVERFLOW	M,T,I
4	ILLEGAL E-UNIT DISPATCH (GATE FAULT)	NONE
5	STOREBACK EXCEPTION	MCWD
6	NAME TRACE TRAP	T,I
7	LOGICAL READ TRACE TRAP	T,I AND DES
8	LOGICAL WRITE TRACE TRAP	T,I AND DES
9	UID READ DEREFERENCE TRAP	DES
10	UID WRITE DEREFERENCE TRAP	DES
11	PROTECTION CACHE MISS	NONE
12	PROTECTION VIOLATION	MCWD
13	PAGE CROSSING INTERRUPT	NONE
14	LAT	NONE
15	WRITE LAT	NONE
16	MEMORY REFERENCE REPEAT	NONE
17	EGG TIMER OVERFLOW	A,M,T,I
18	E-BOX STACK UNDERFLOW	A,M,T,I
19	NON-FATAL MEMORY ERROR	NONE
20	INTERVAL TIMER OVERFLOW	NONE
21	IPM INTERRUPT	NONE
22	S-OP TRACE TRAP	NONE
23	ILLEGAL S-OP	NONE
24	MICROINSTRUCTION TRACE TRAP	NONE
25	NON-PRESENT MICROINSTRUCTION	NONE
26	INSTRUCTION PREFETCH IS HUNG	NONE
27	F-BOX STACK UNDERFLOW	NONE
28	MICROINSTRUCTION BREAK POINT TRACE TRAP	T,I AND MCWD
29	MISS ON NAME CACHE LOAD OR READ REGISTER	NONE

FIG. 247

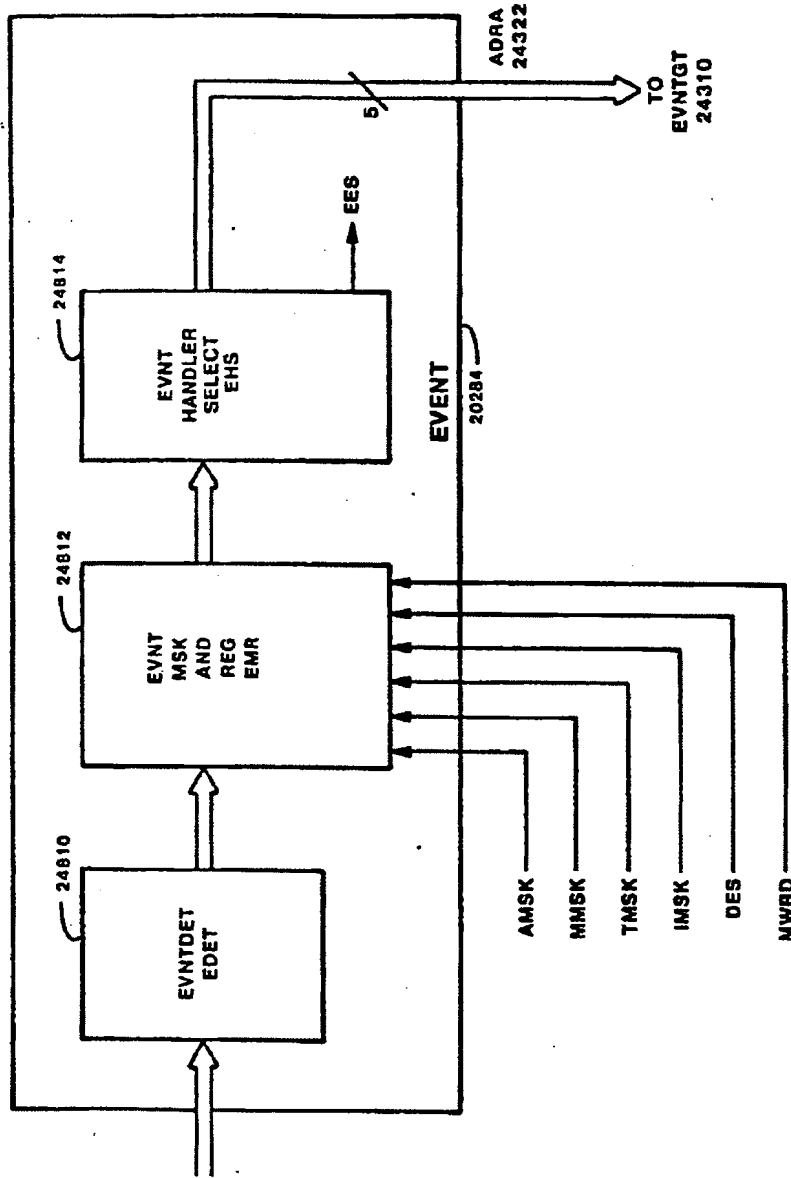


FIG. 248

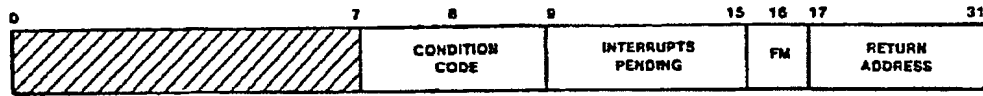
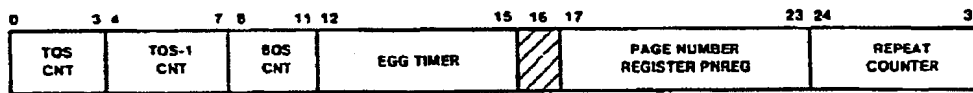
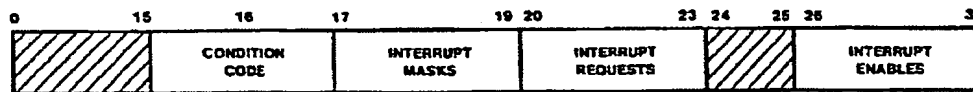


FIG. 251



MCWO



MCW1

FIG. 252

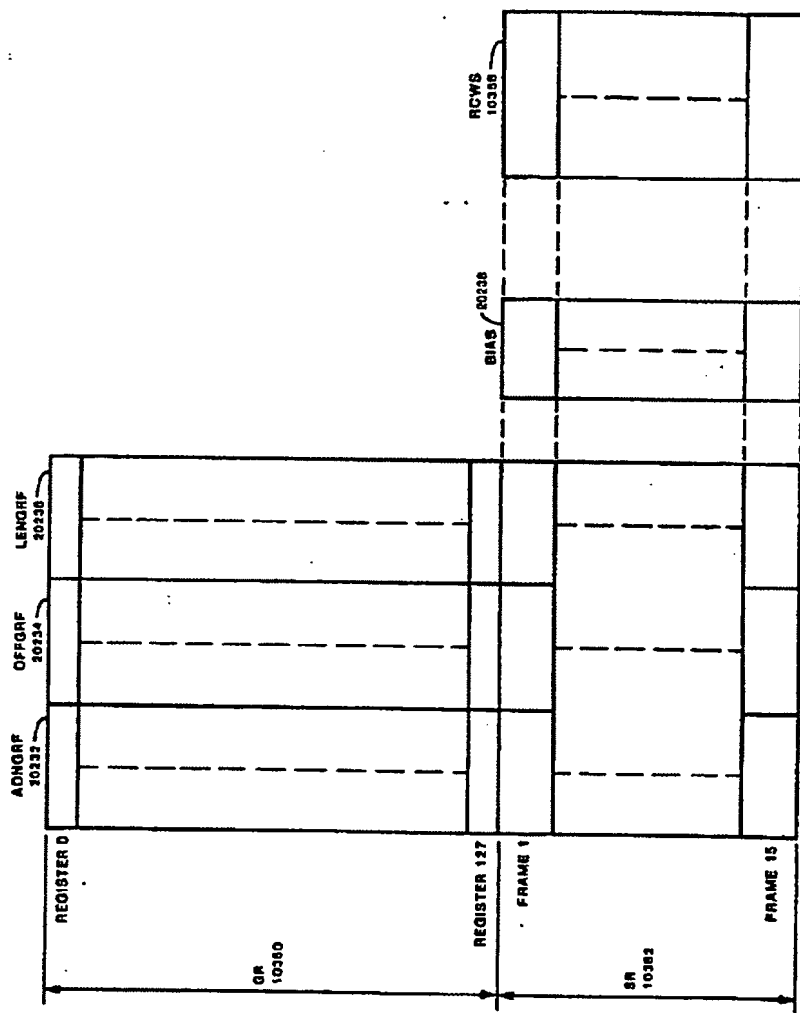


FIG. 253

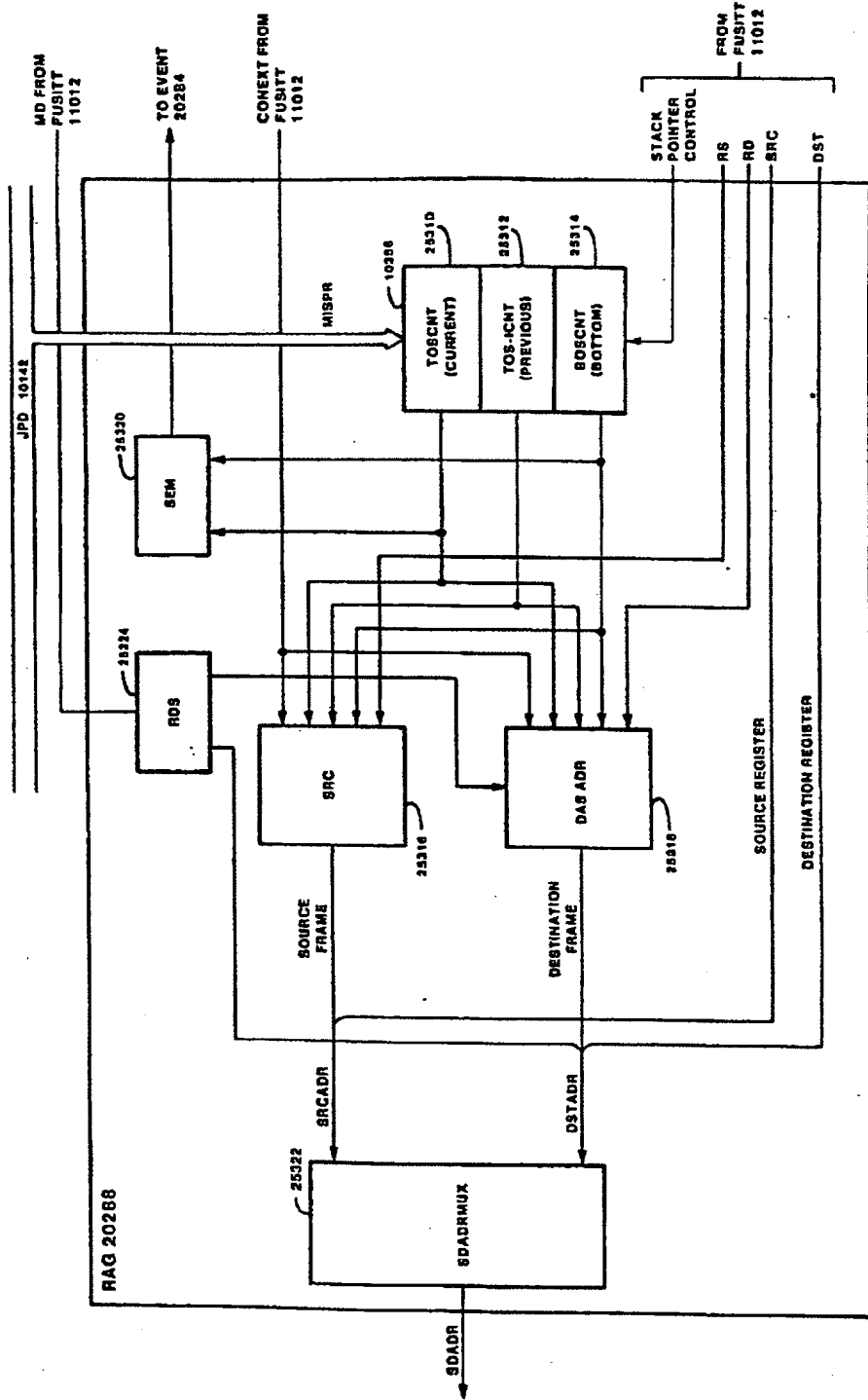


FIG. 253A

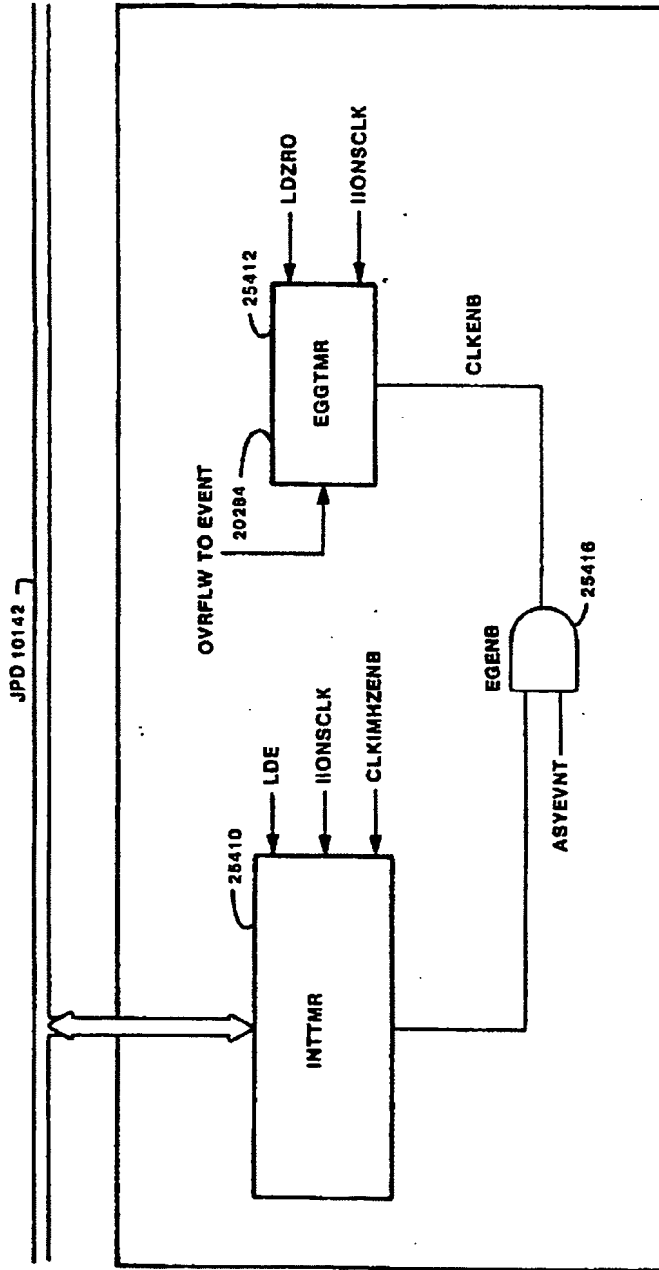


FIG. 254

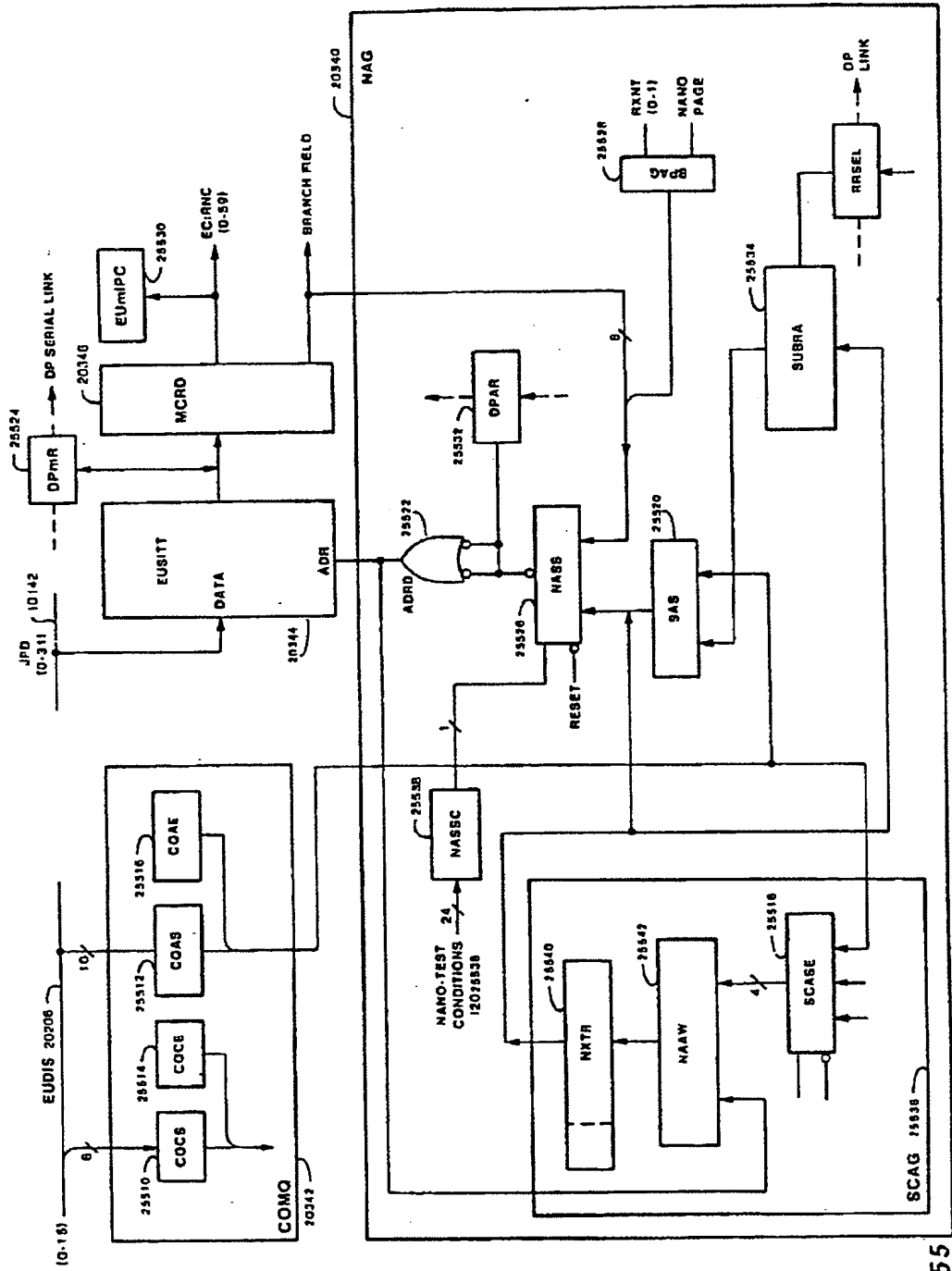


FIG. 255

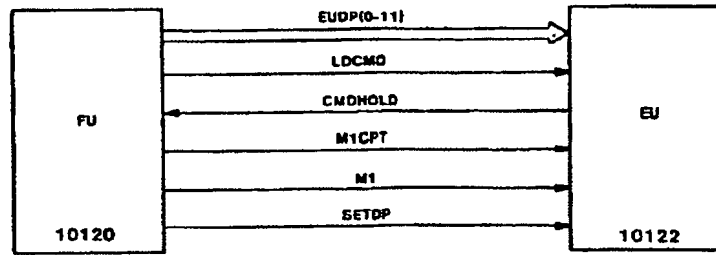


FIG. 260

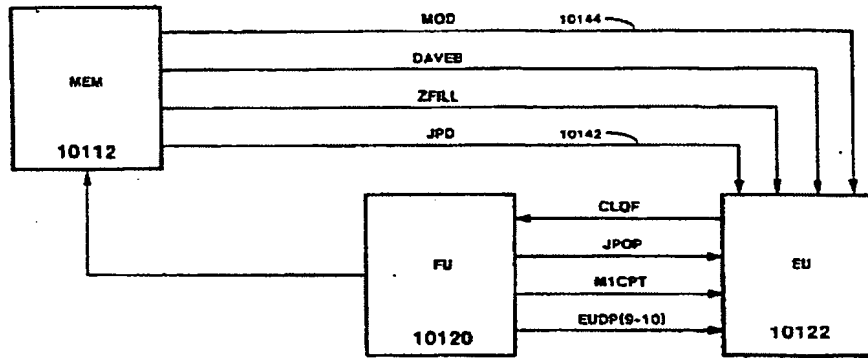


FIG. 261

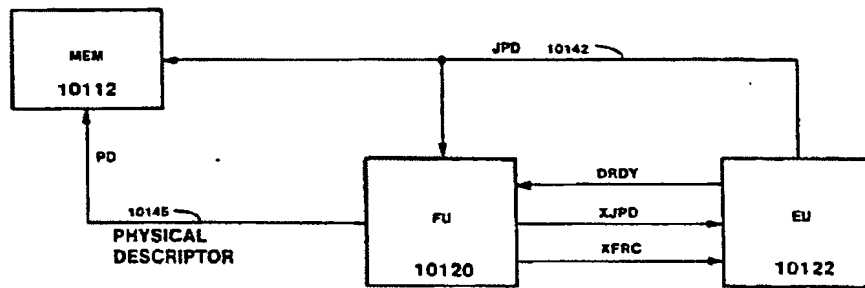


FIG. 262

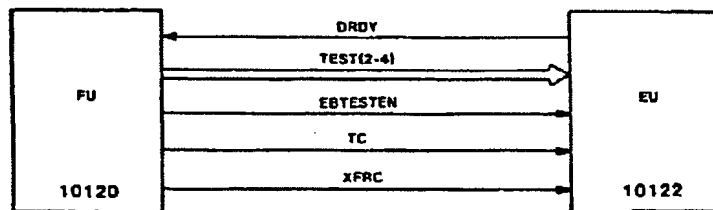


FIG. 263

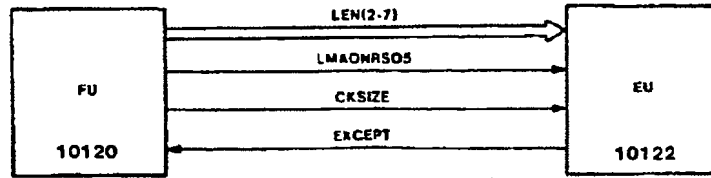


FIG. 264

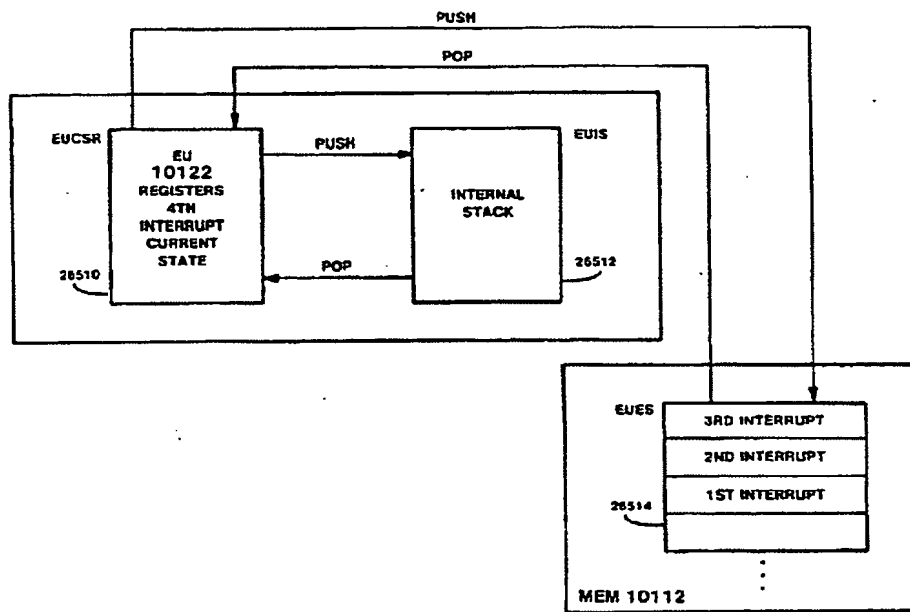


FIG. 265

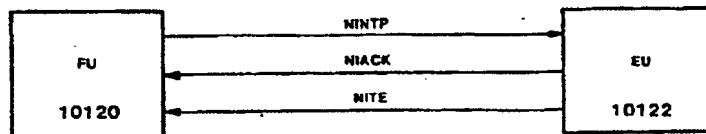


FIG. 266

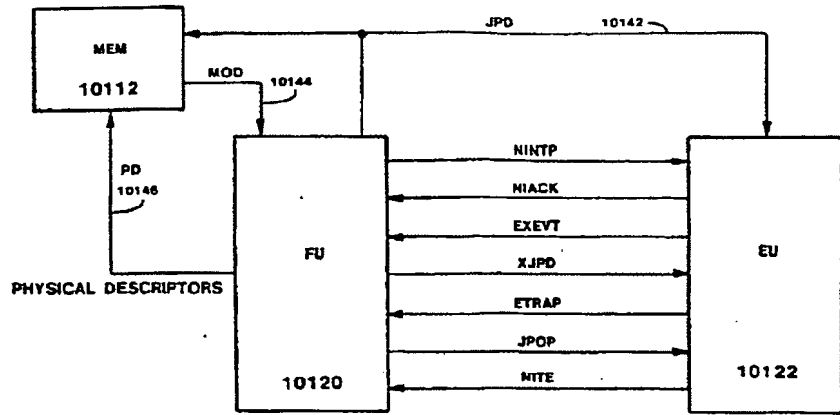


FIG. 267

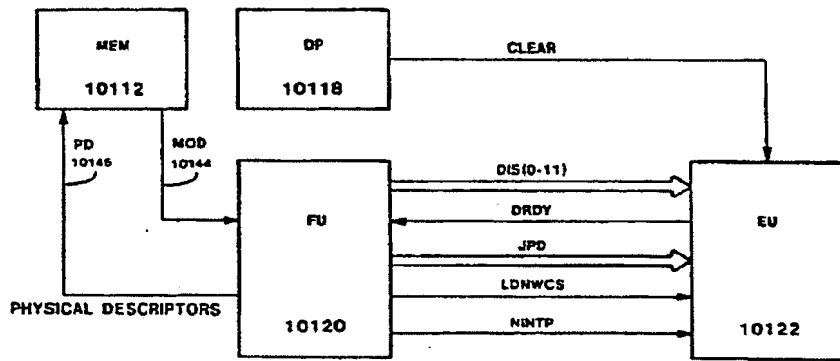


FIG. 268

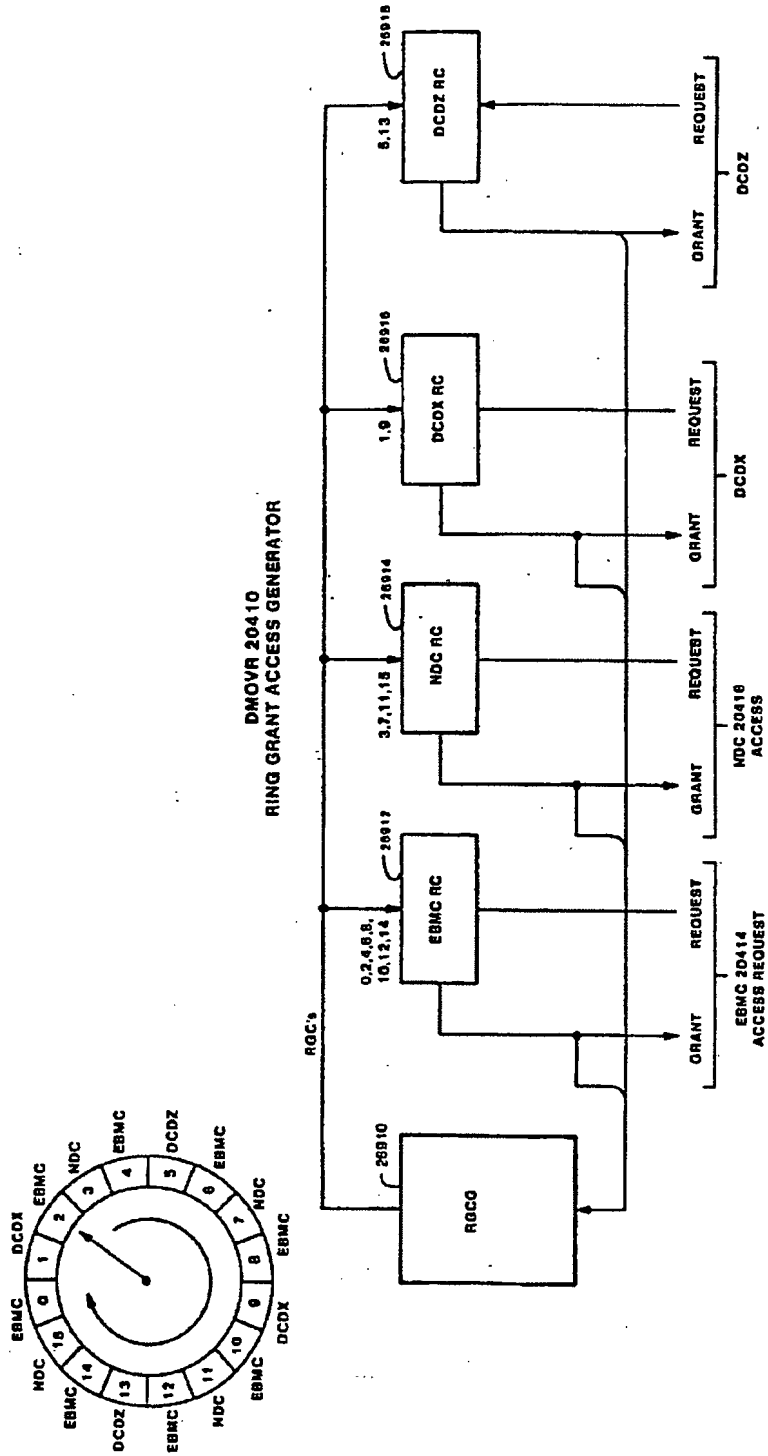


FIG 269

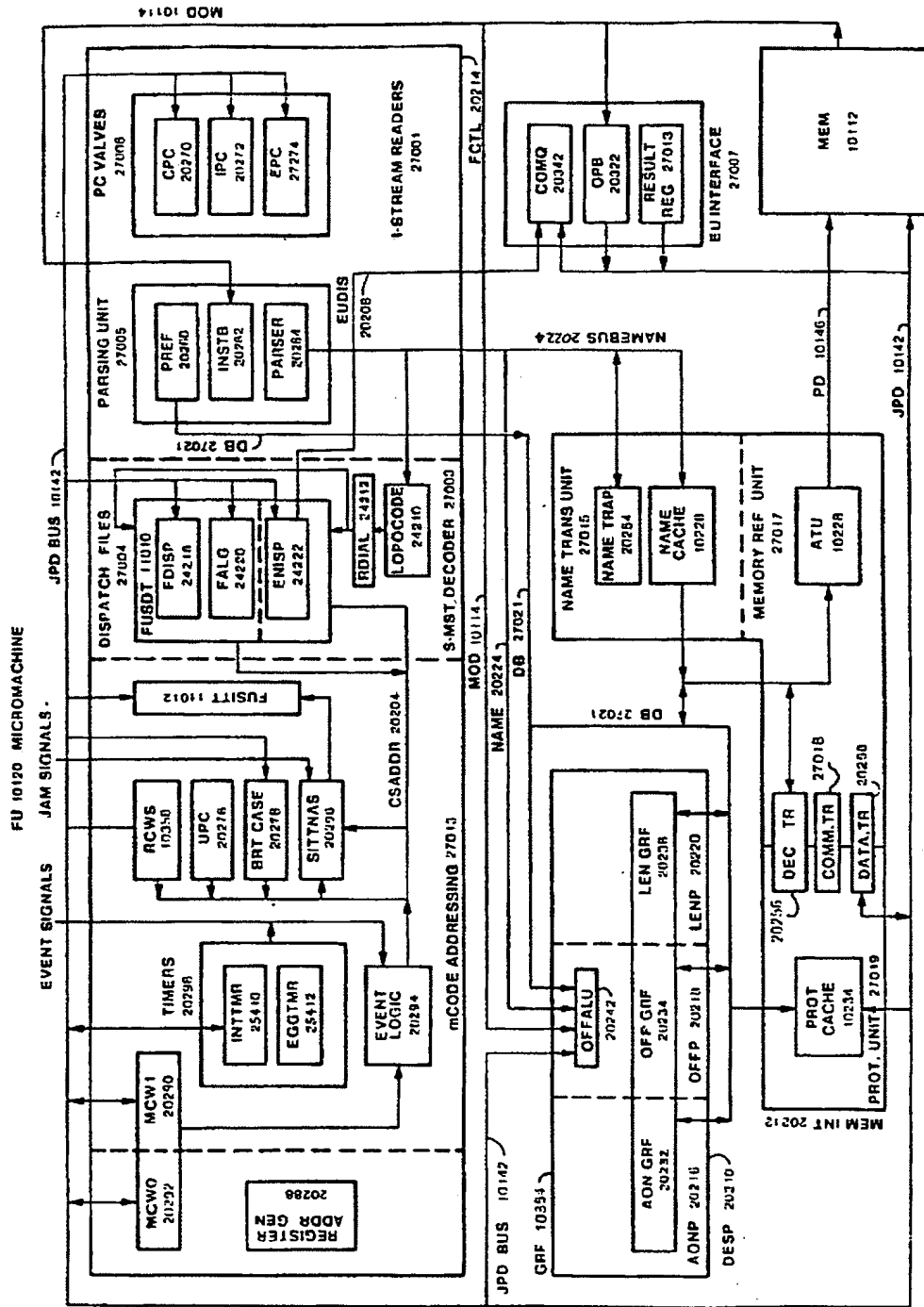


FIG 270

LOGICAL DESCRIPTOR DETAIL

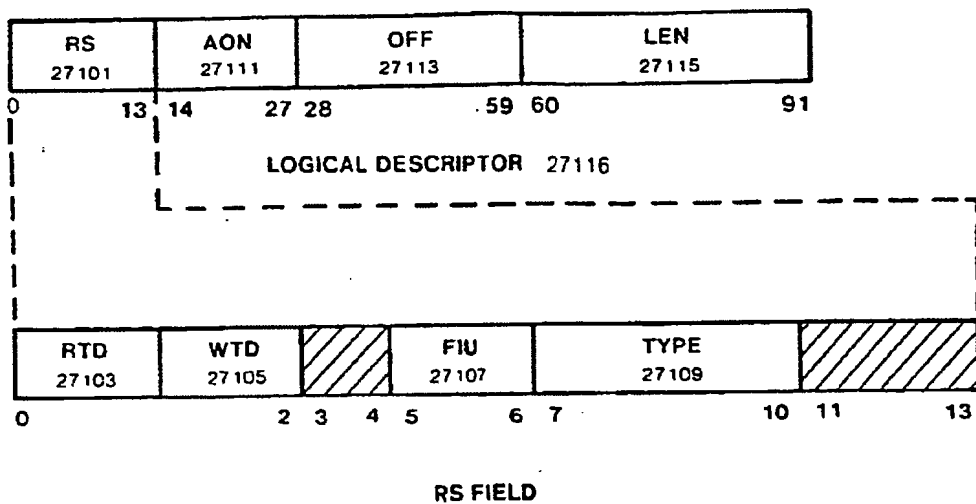


FIG. 271

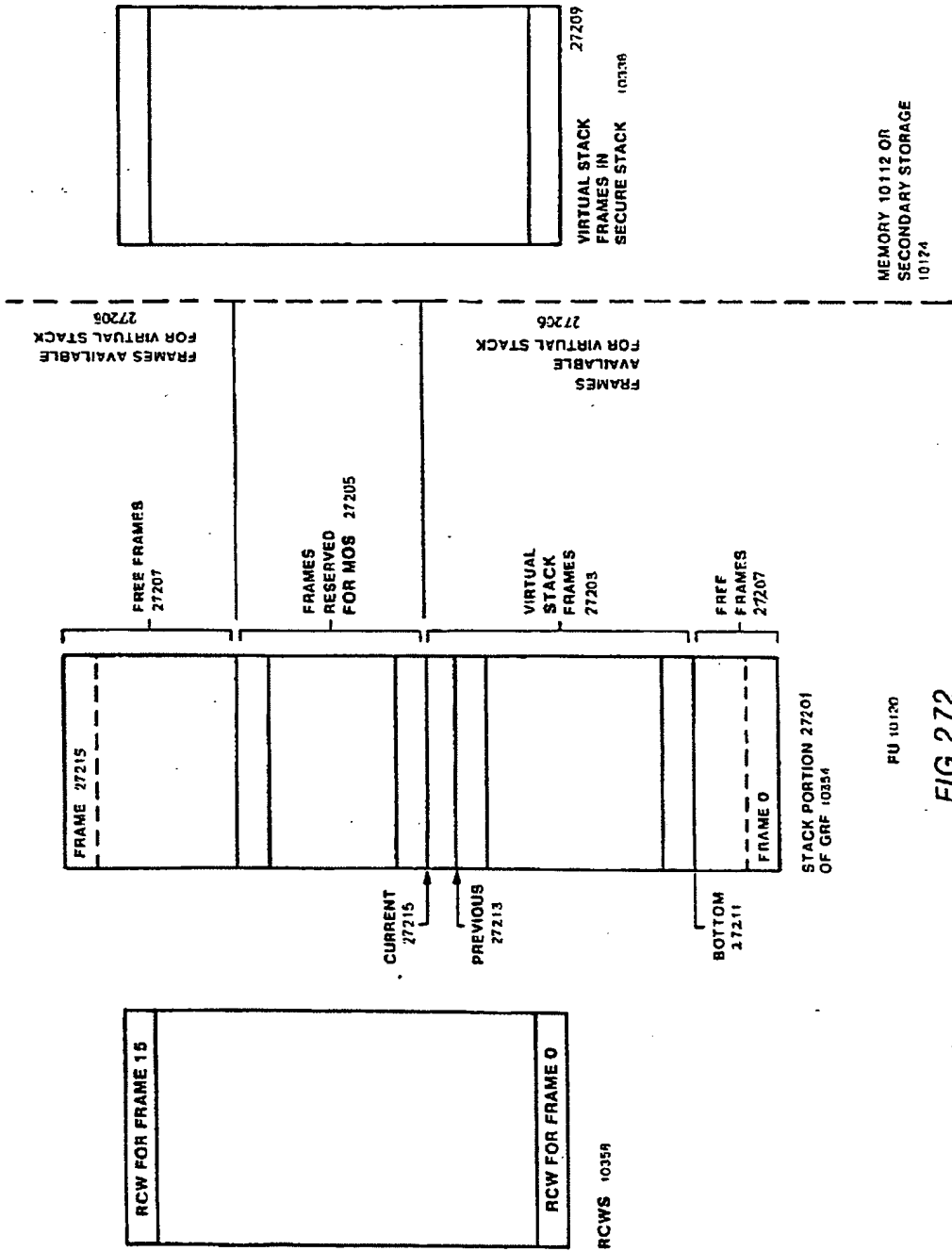


FIG 272

EP 0 067 556 B1

STRUCTURES CONTROLLING EVENT INVOCATION

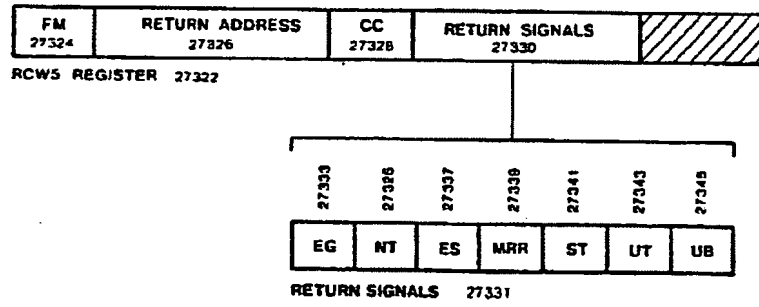
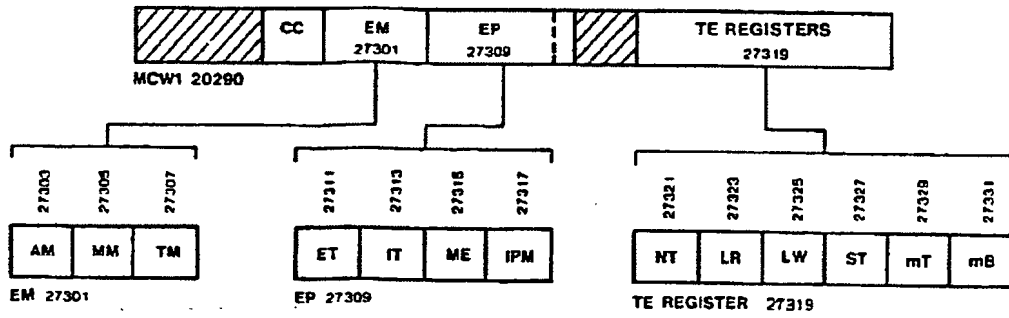


FIG. 273

POINTER FORMATS

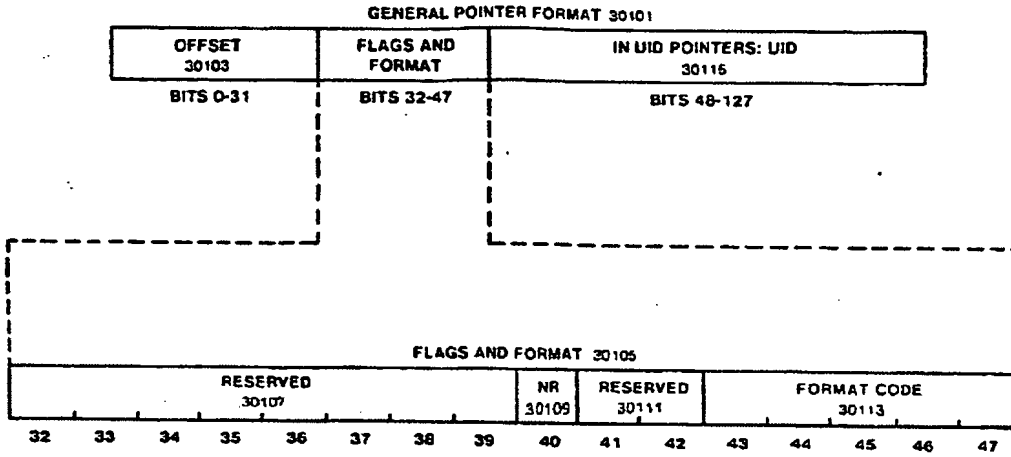


FIG. 301

ASSOCIATED ADDRESS TABLE

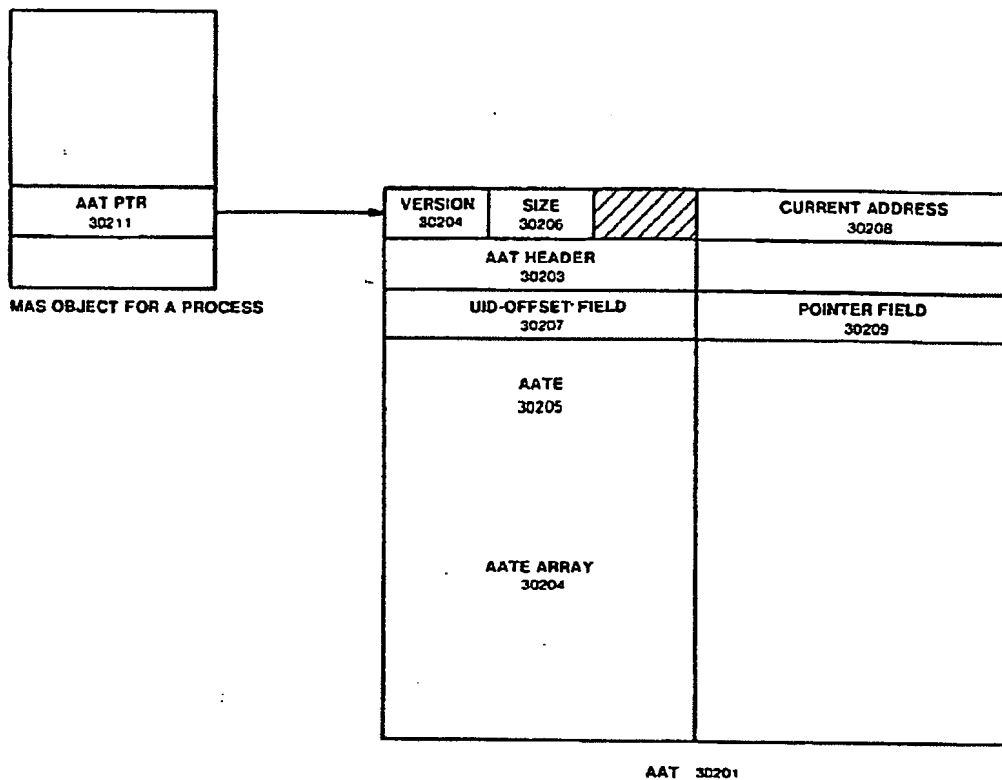


FIG. 302

NAMESPACE OVERVIEW OF A PROCEDURE OBJECT

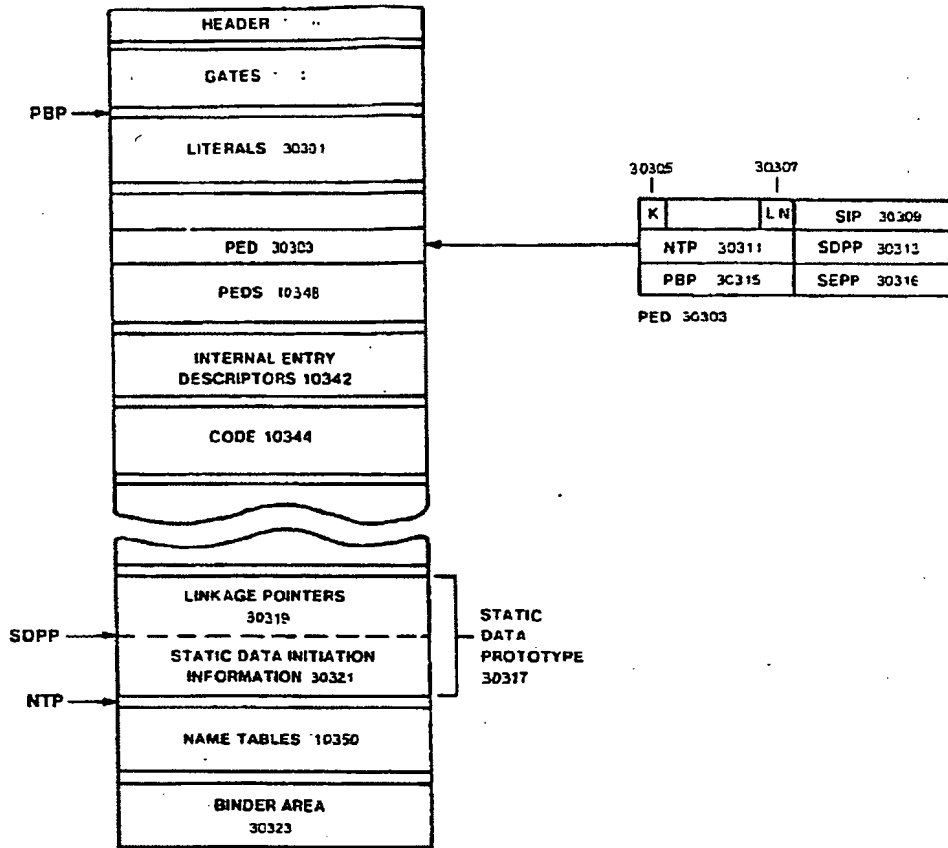


FIG. 303

EP 0 067 556 B1

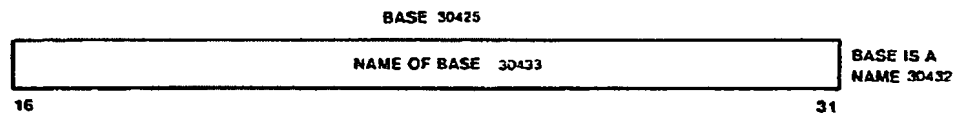
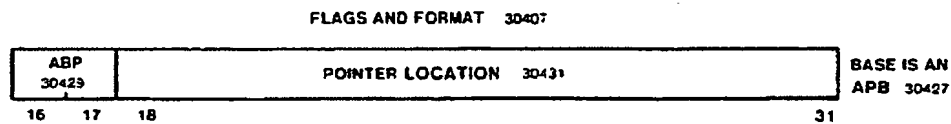
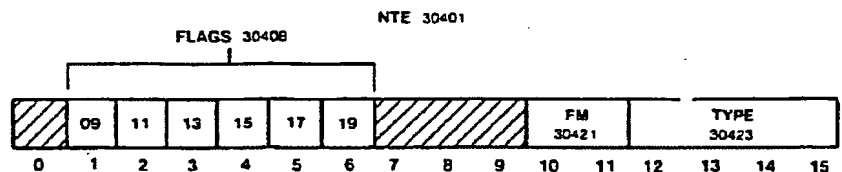
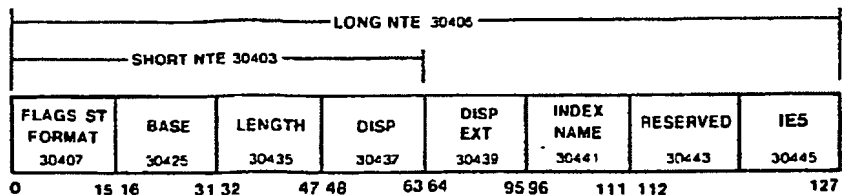
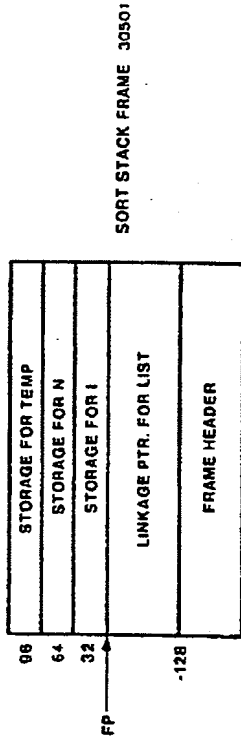


FIG. 304

NAME RESOLUTION EXAMPLE



30407	30426	30436	30437	30438	30441	30446
FLAGS SEE BELOW	A B P	PTR, DISP 1	LENGTH 32	DISP 0	DISP EXT: UNUSED	INDEX NAME: I'S NAMES

NTE FOR LIST (I) 30502 : FLAGS SET: LONG NTE 30409
 BASE IS INDIRECT 30415
 ARRAY 30417
 ABP 00 (FP)

A B P	UNUSED	LENGTH: 32	DISP. 0
-------------	--------	---------------	------------

NTE FOR I 30403 : FLAGS SET: NONE; ABP: 00 (FP)

FIG. 305

NAME CACHE REGISTERS

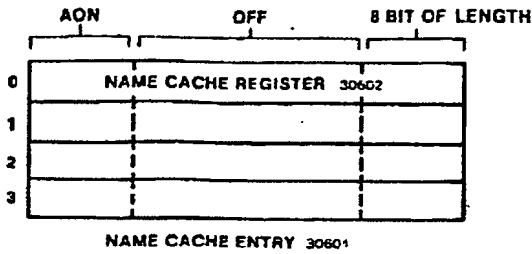


FIG. 306

TRANSLATING S-INTERPRETER UIDS TO DIALECT NUMBERS

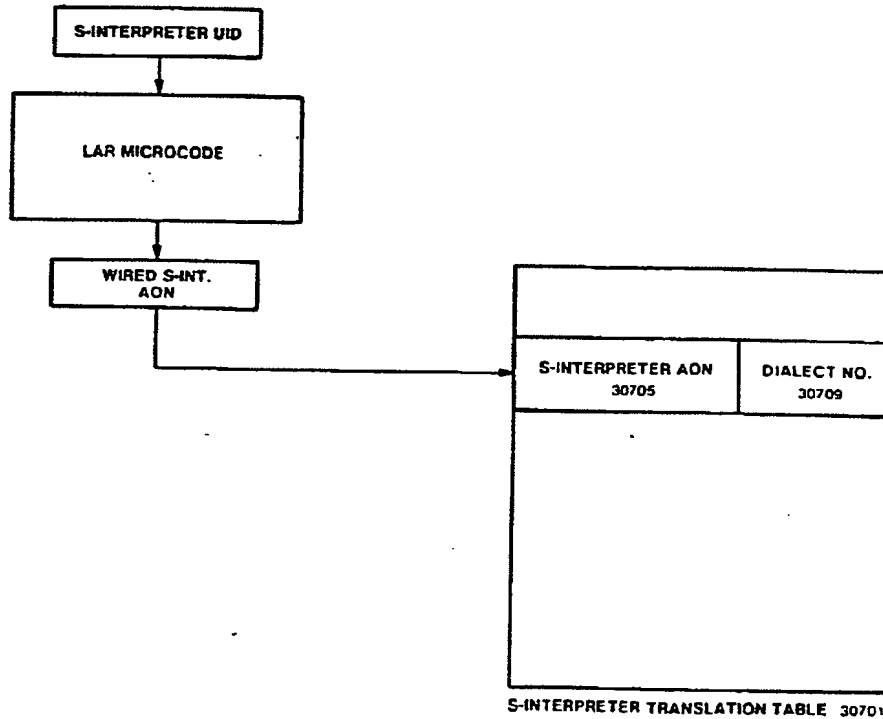


FIG. 307

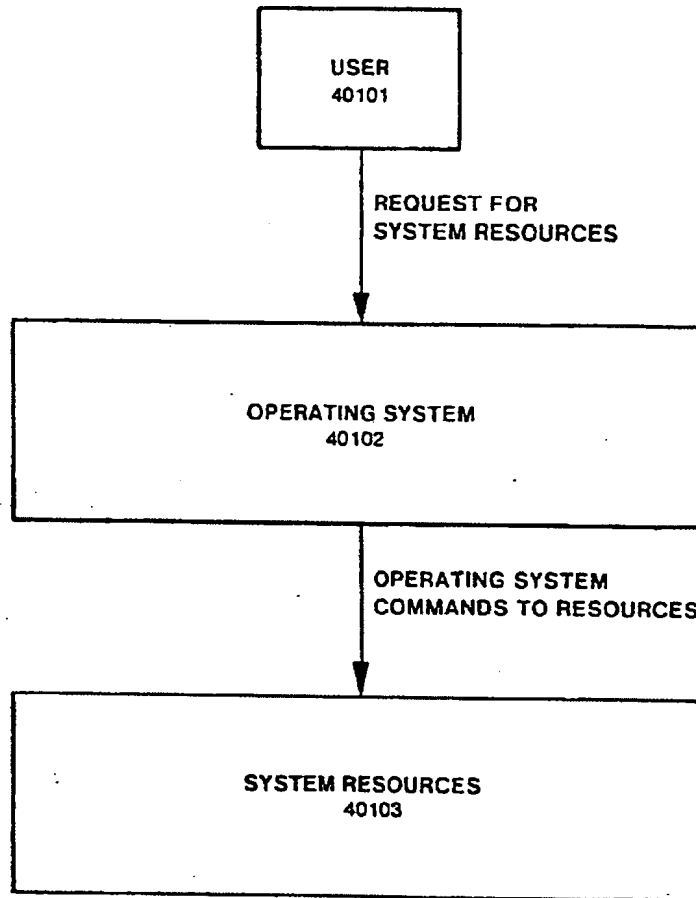


FIG 401

MULTIPROCESS OPERATING SYSTEM

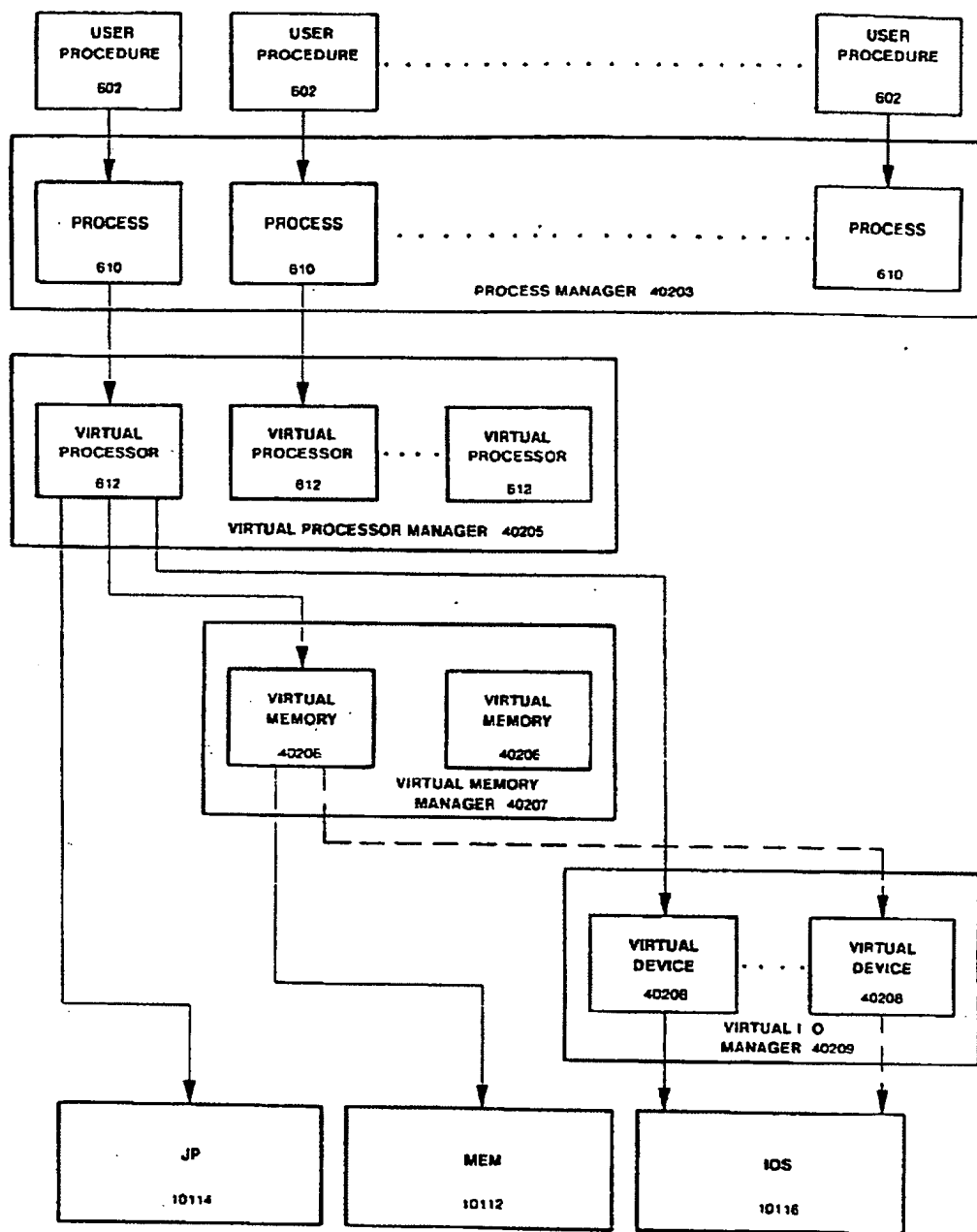


FIG 402

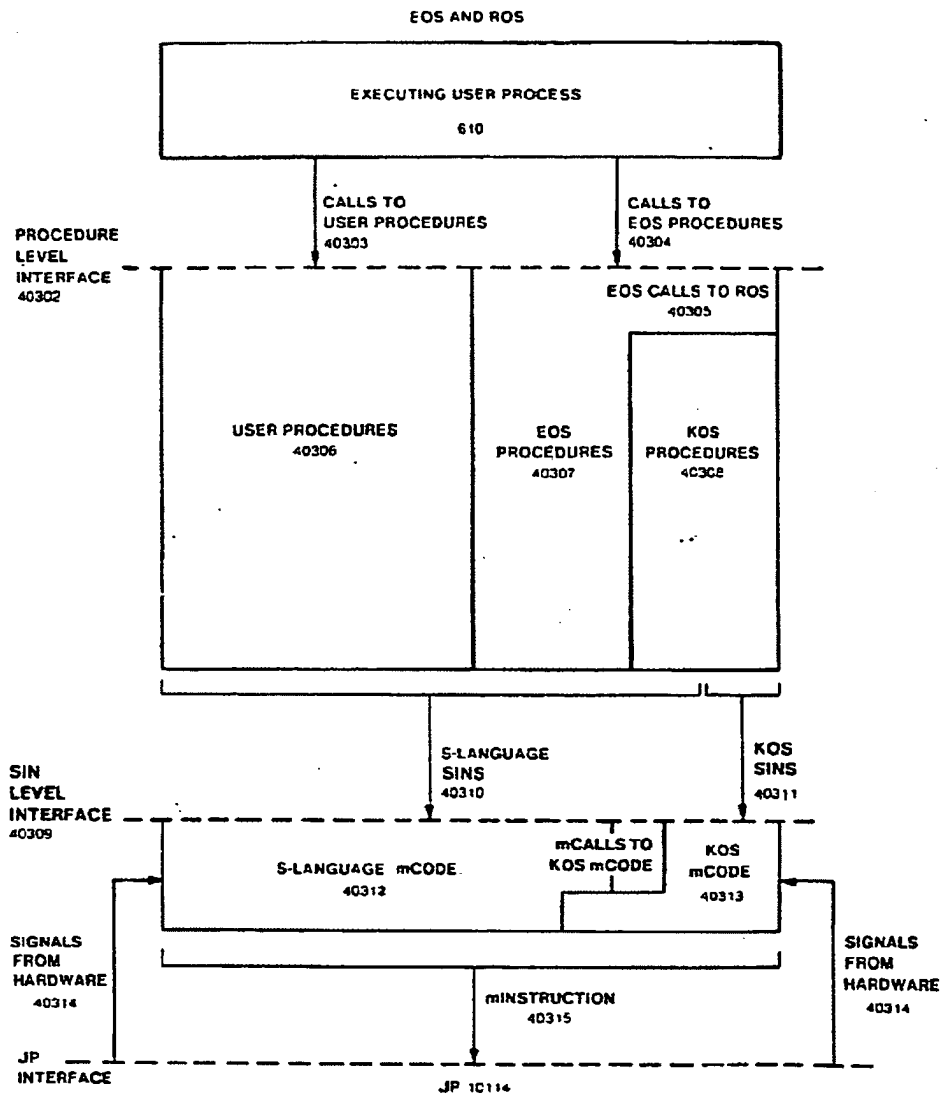


FIG. 403

EOS VIEW OF OBJECTS

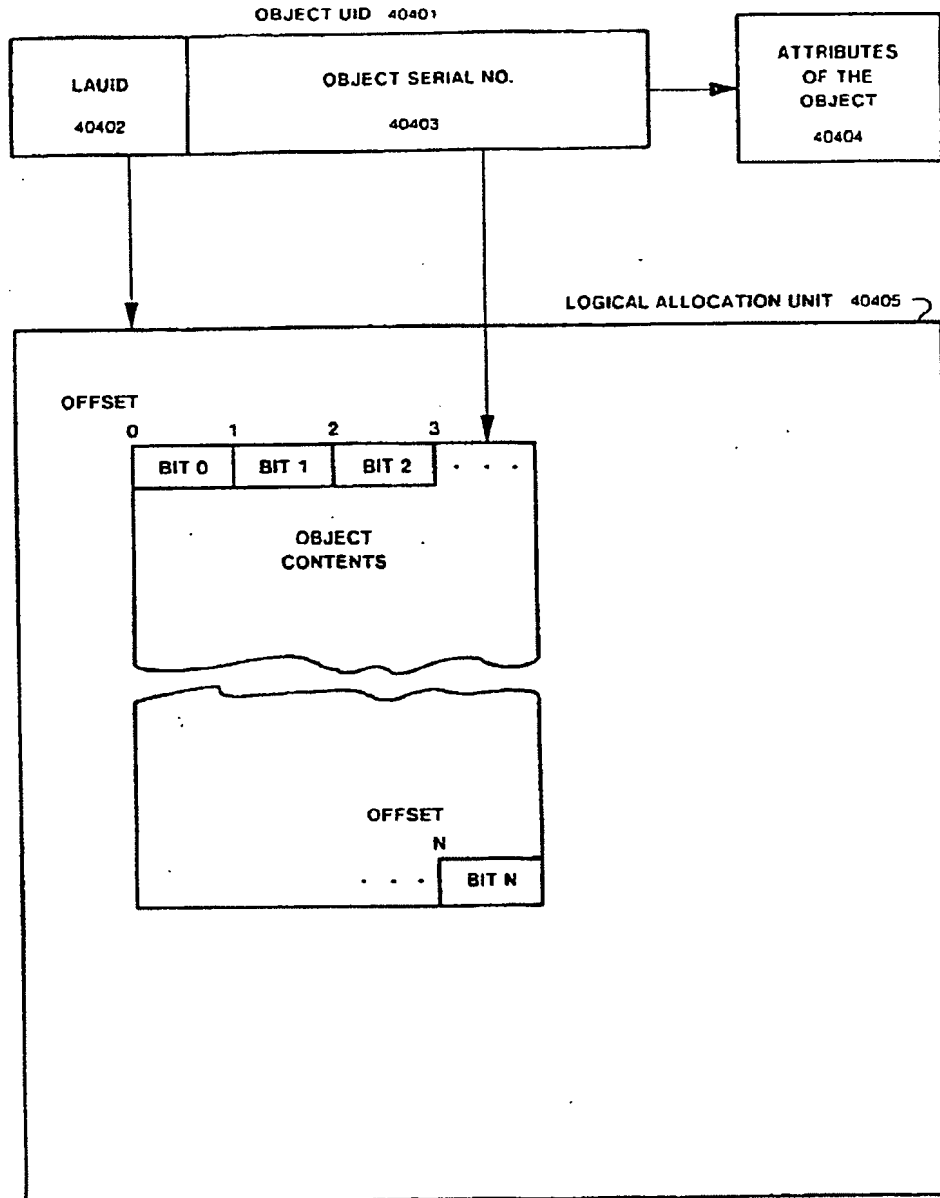


FIG 404

PATHNAME TO UID-OFFSET TRANSLATION

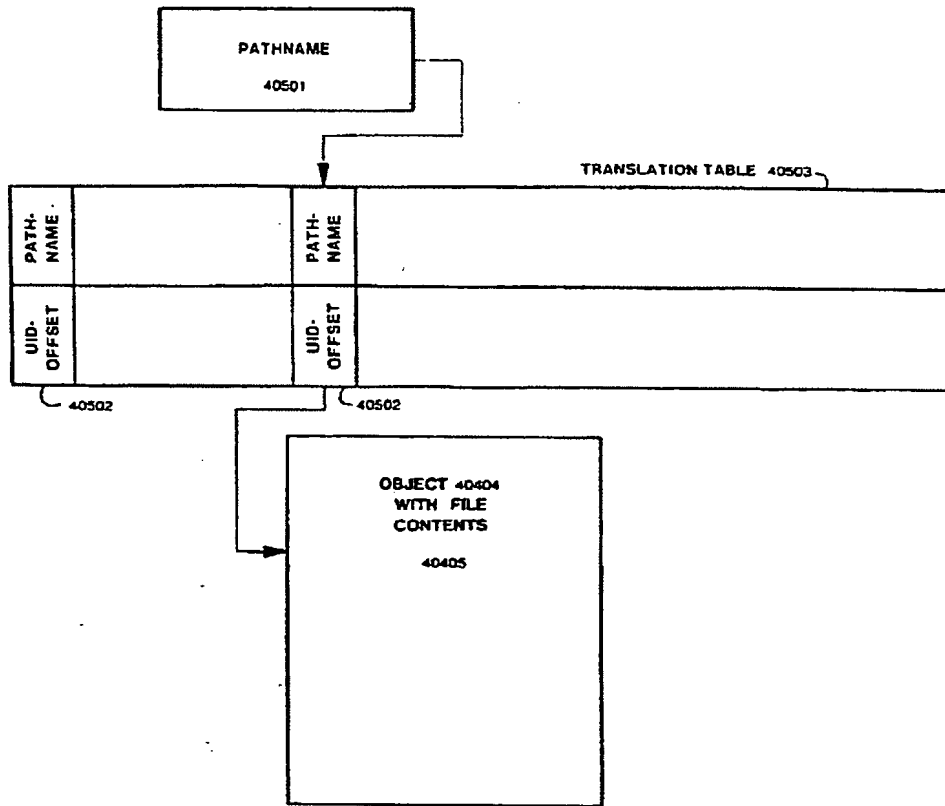


FIG 405

OBJECT UID'S UNIVERSAL IDENTIFIER 01

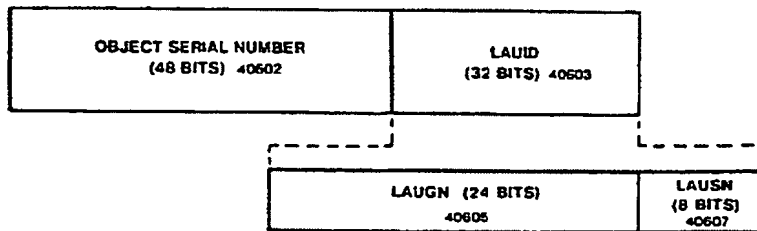


FIG 406

ATU, MHT, AND MEMORY

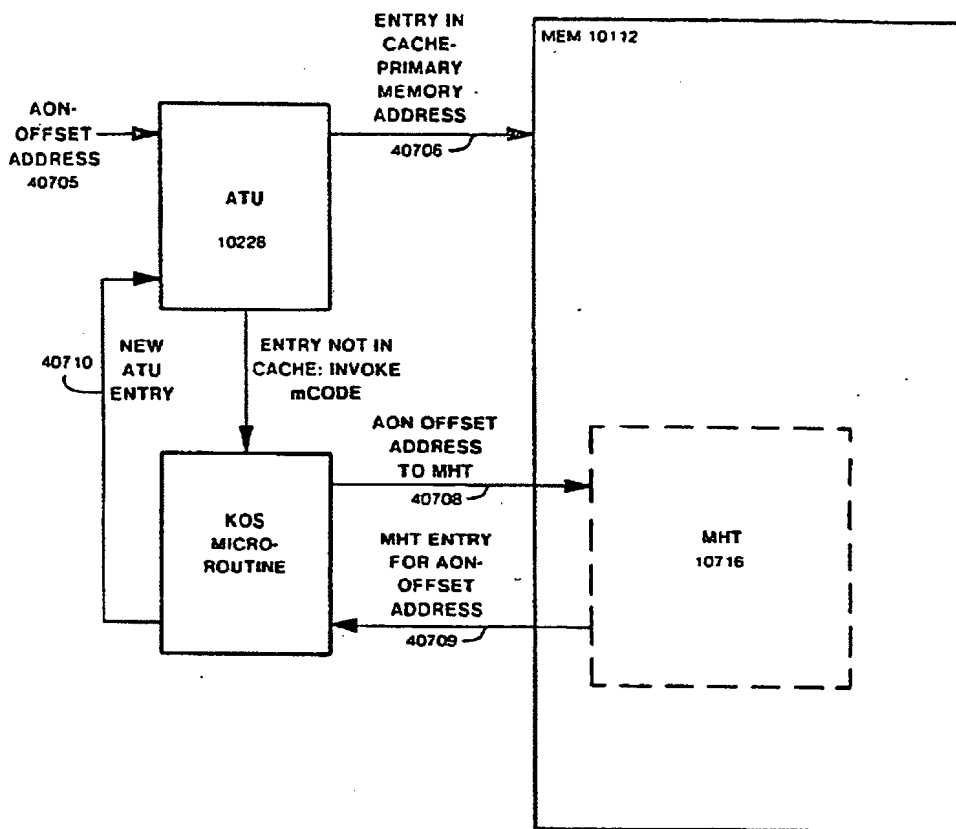


FIG 407

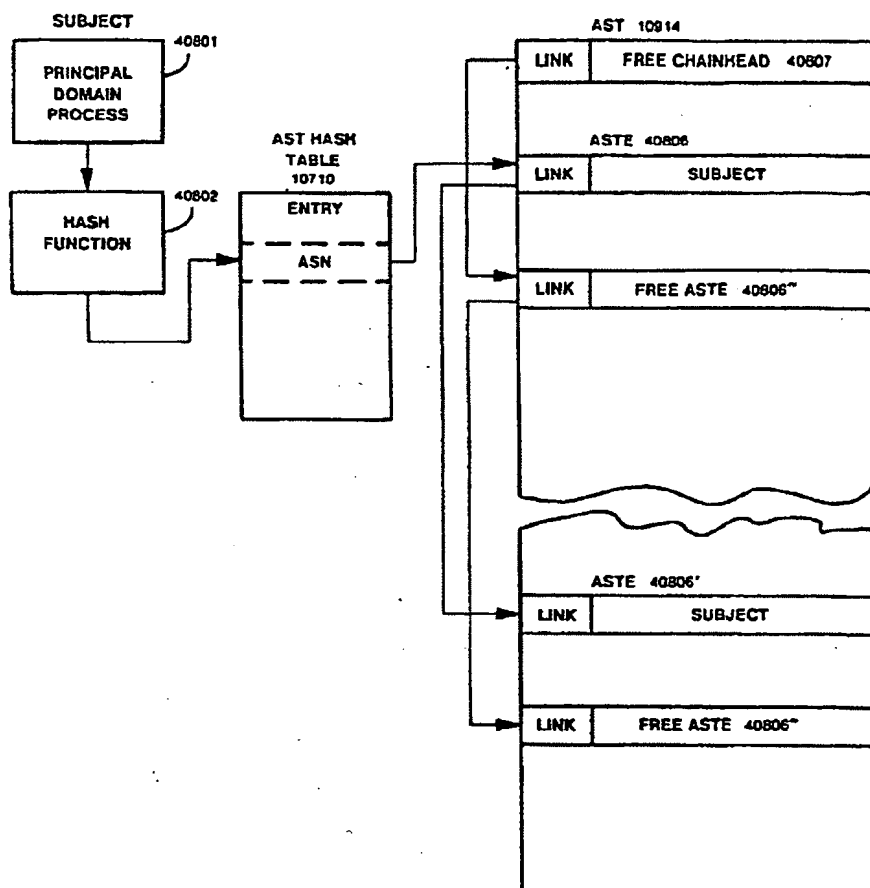


FIG 408

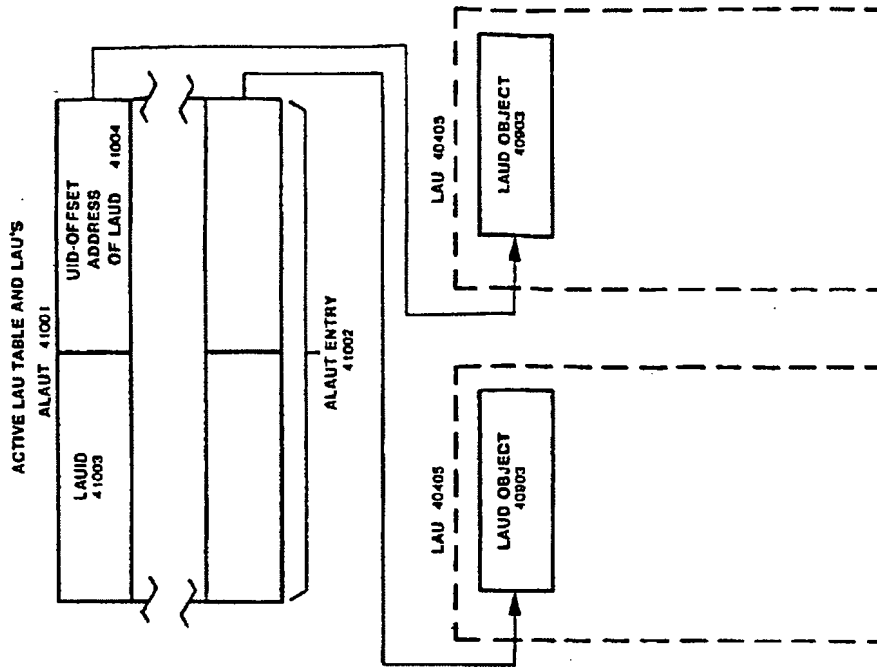


FIG 410

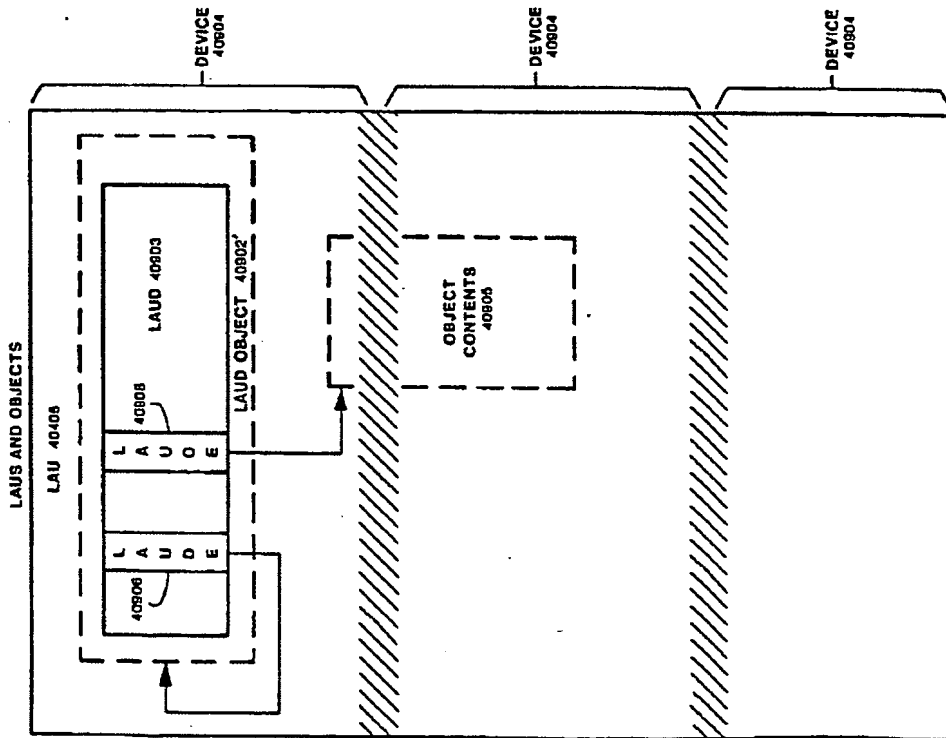


FIG 409

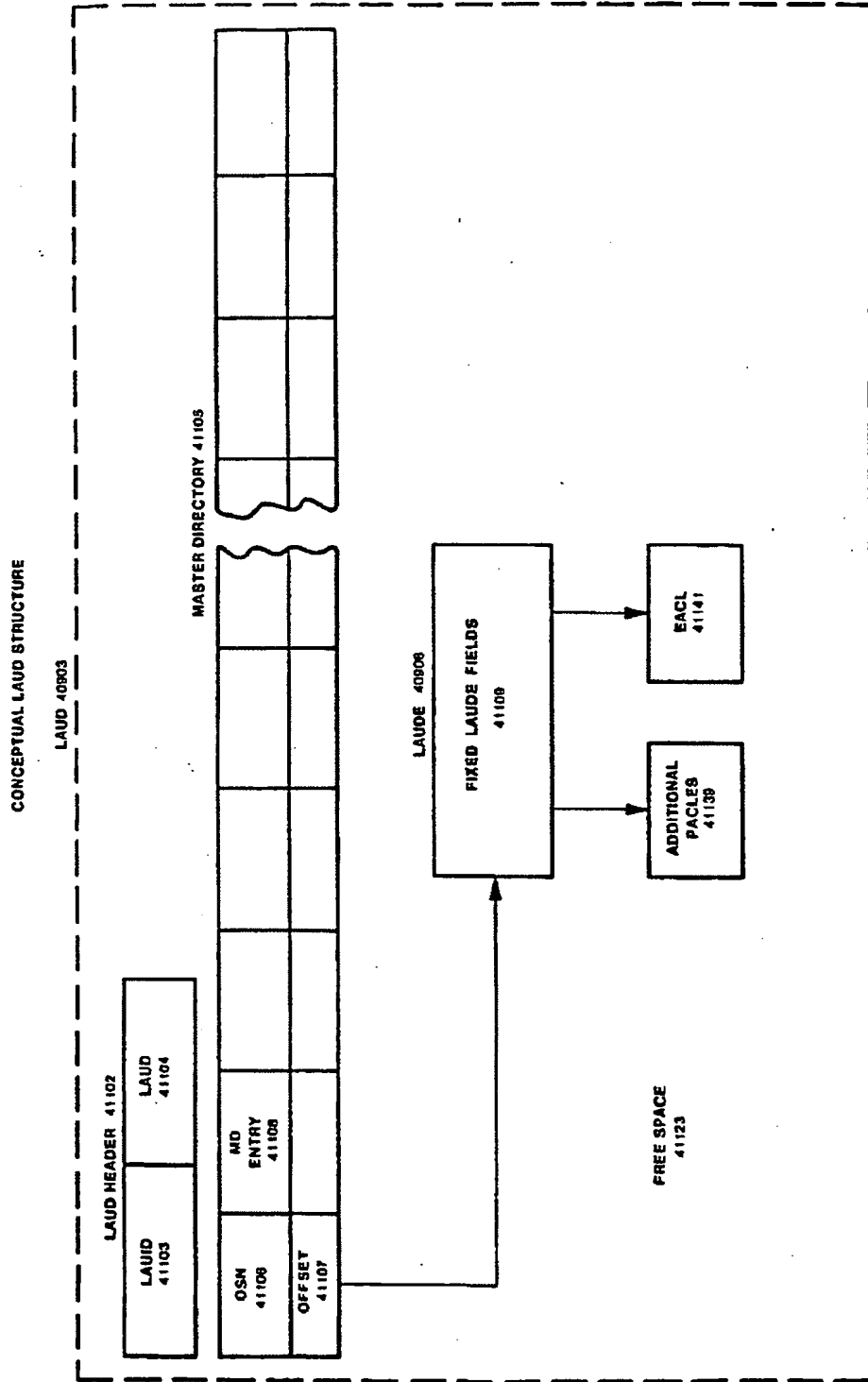
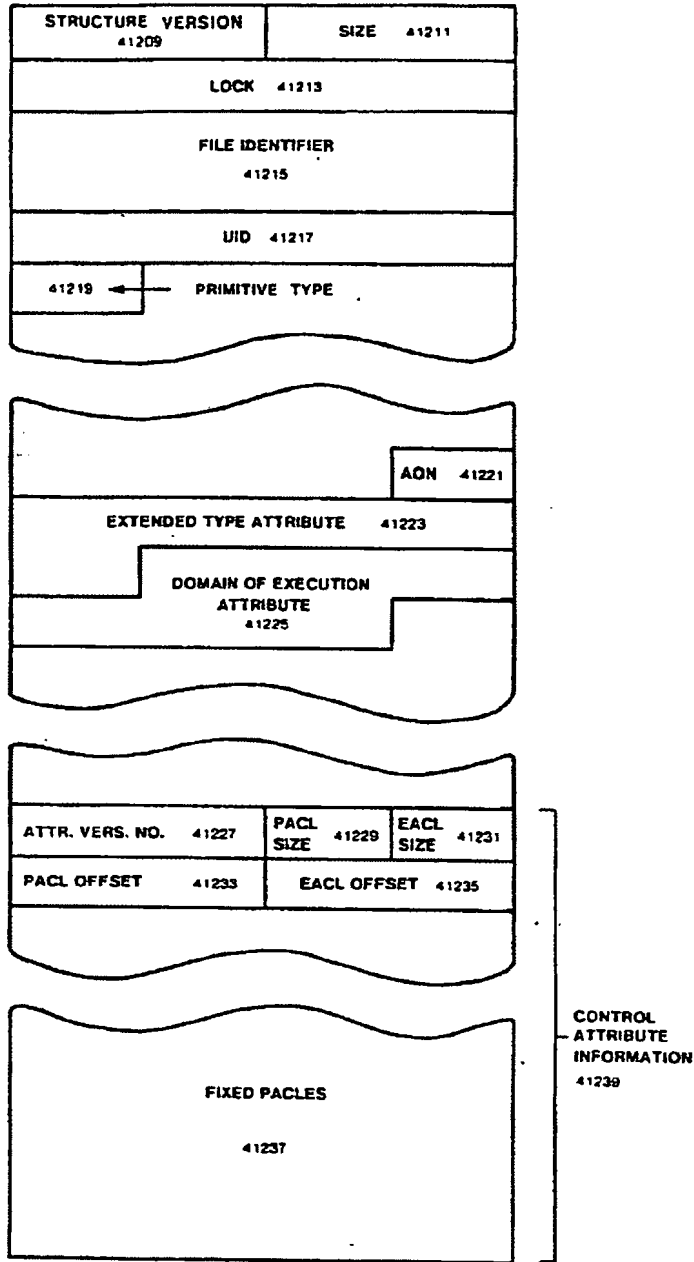


FIG 411

LAUDE DETAIL



LAUDE 40906

FIG. 412

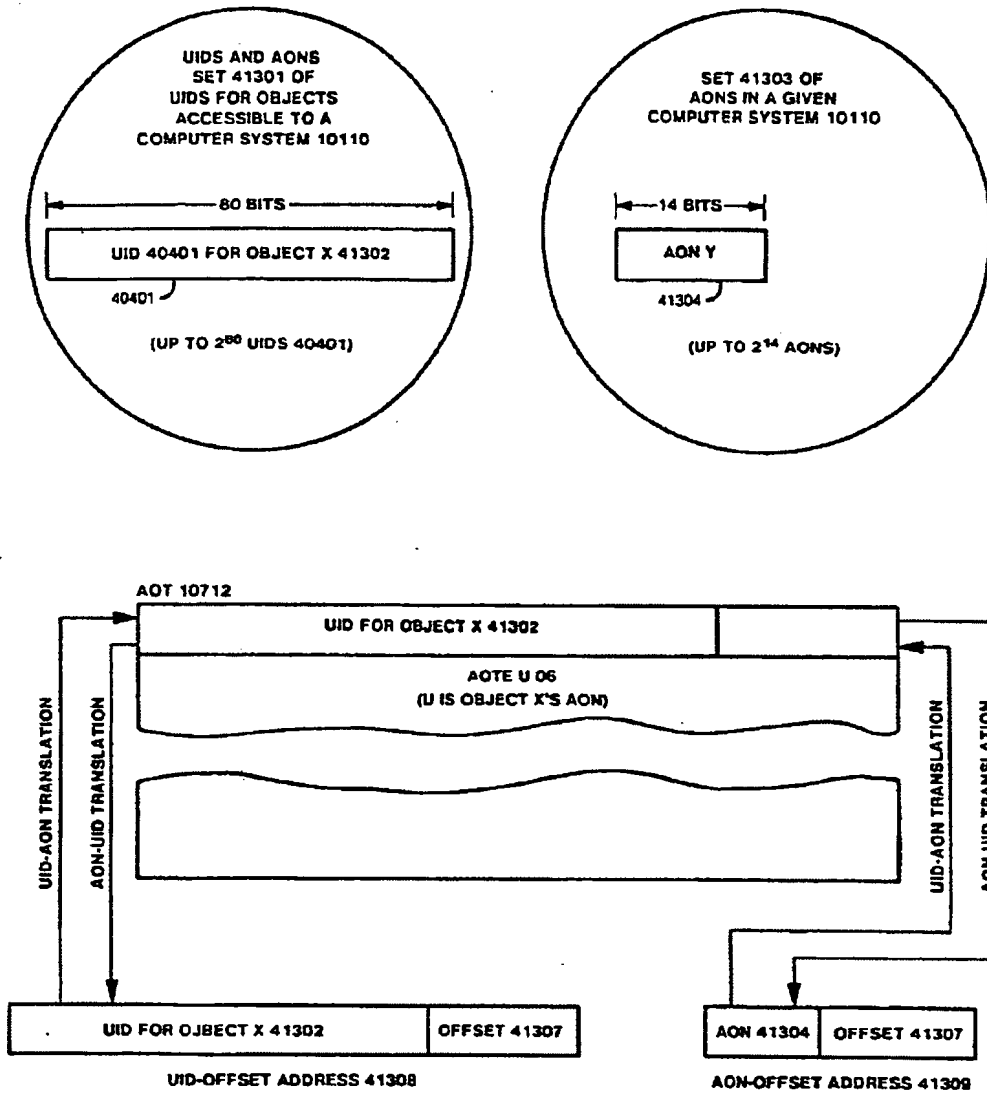


FIG 413

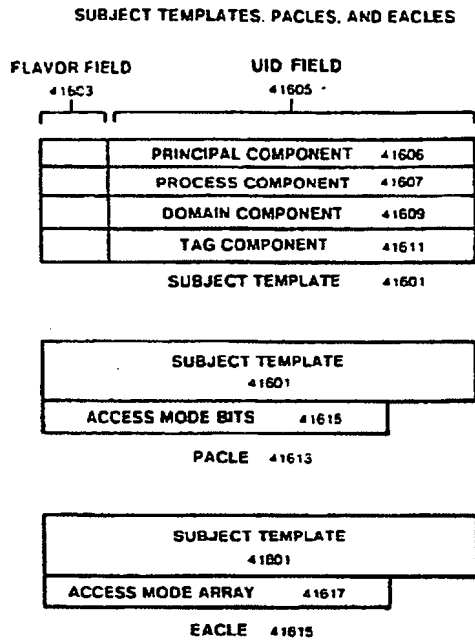


FIG. 416

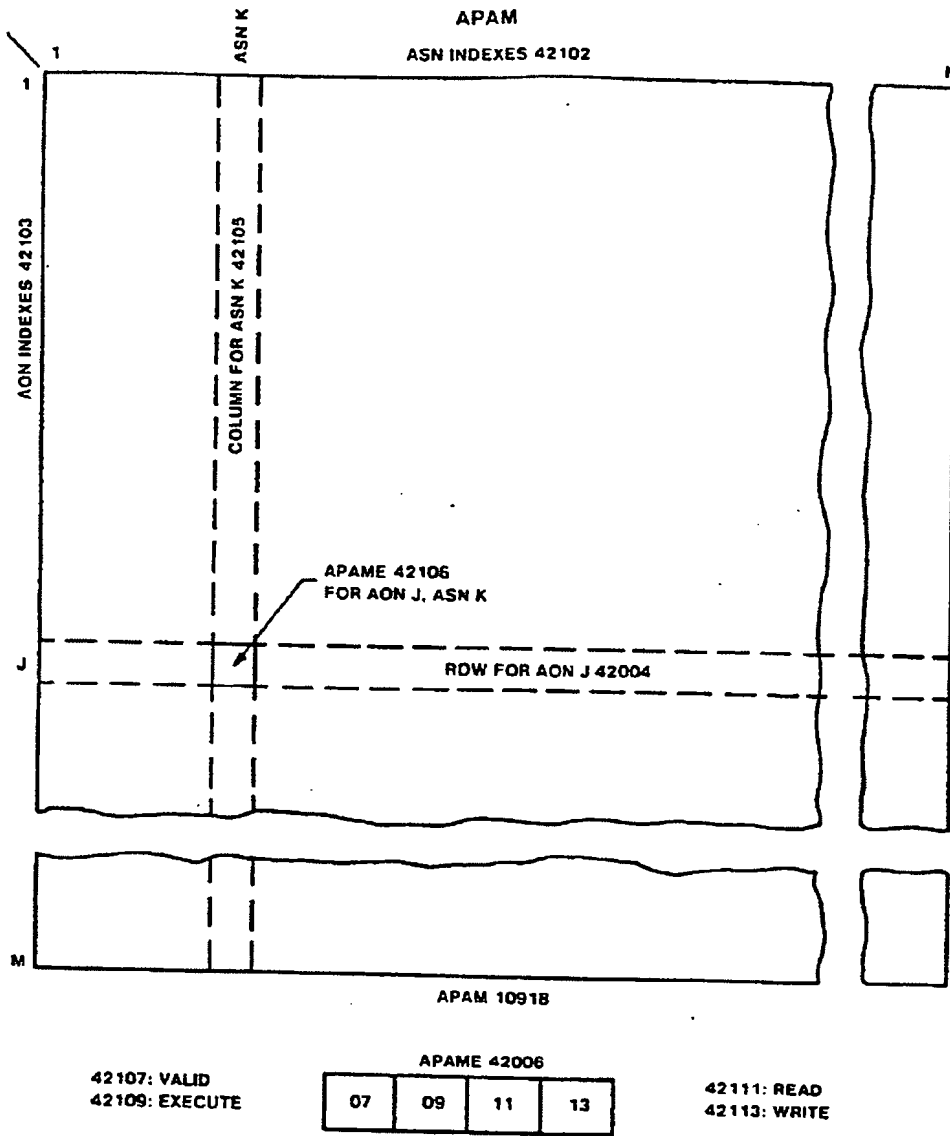


FIG. 421

PRIMITIVE DATA ACCESS CHECKING

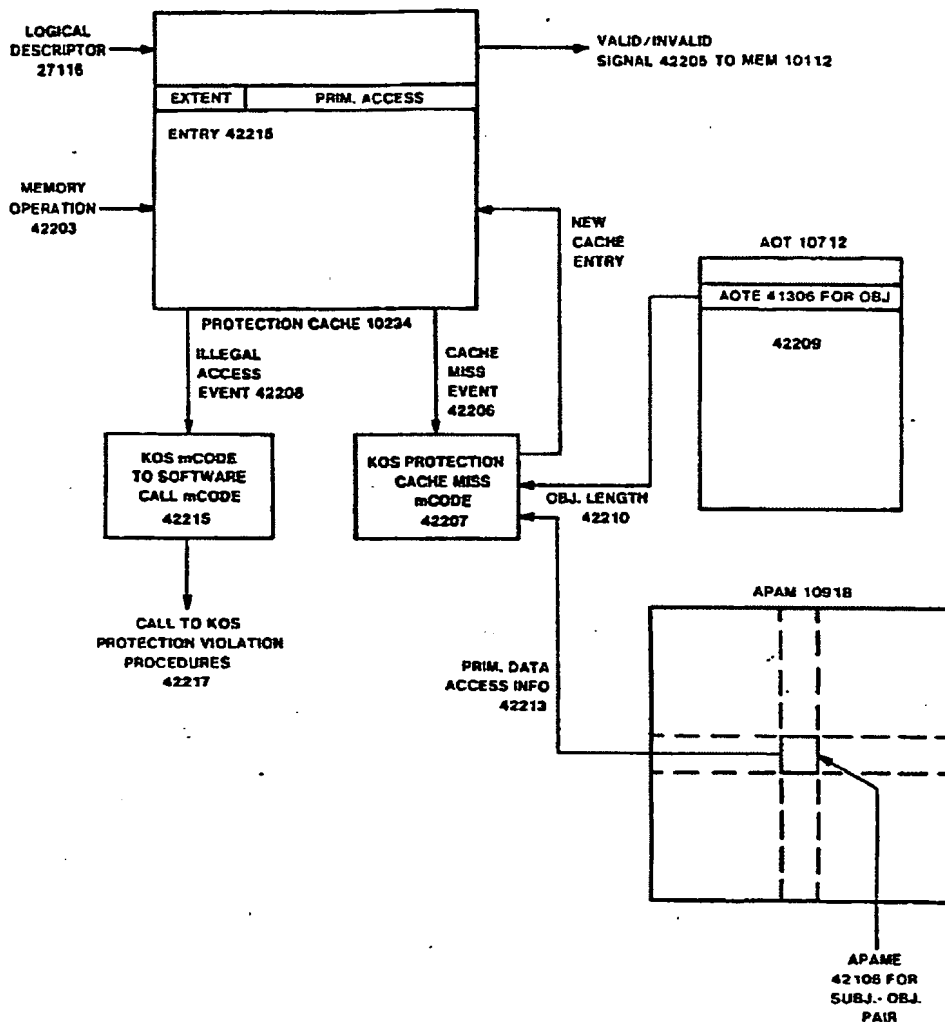


FIG. 422

EVENT COUNTERS AND AWAIT ENTRIES

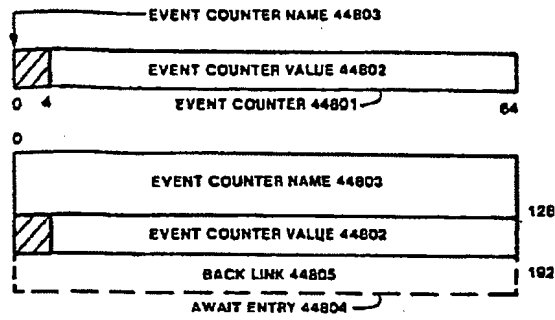


FIG. 448

AWAIT TABLE OVERVIEW

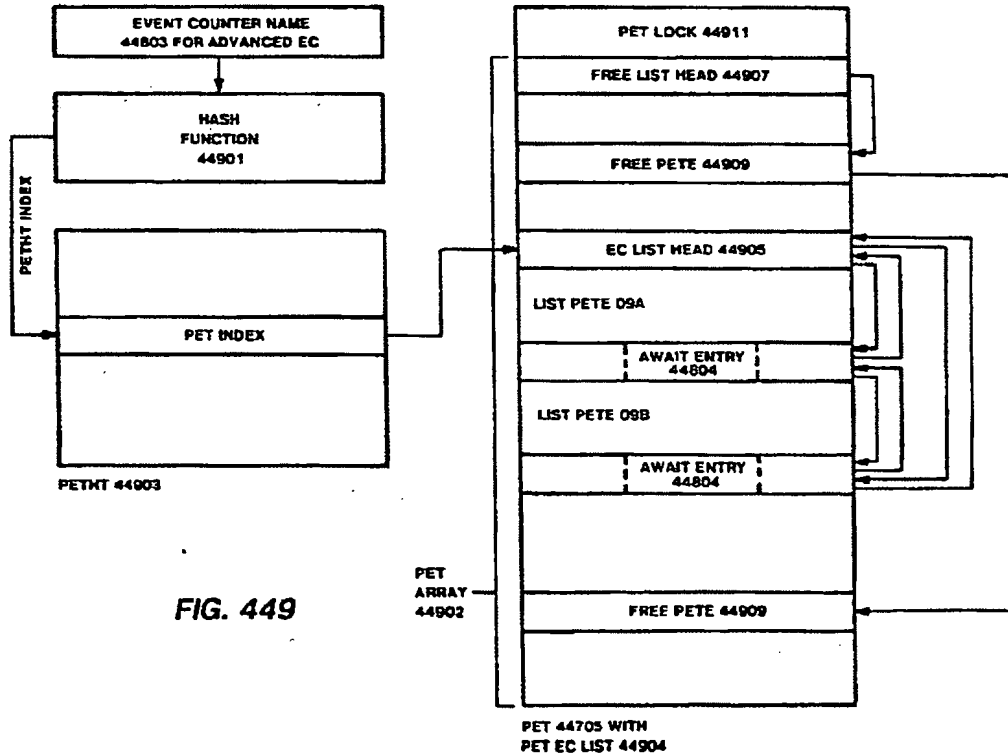


FIG. 449

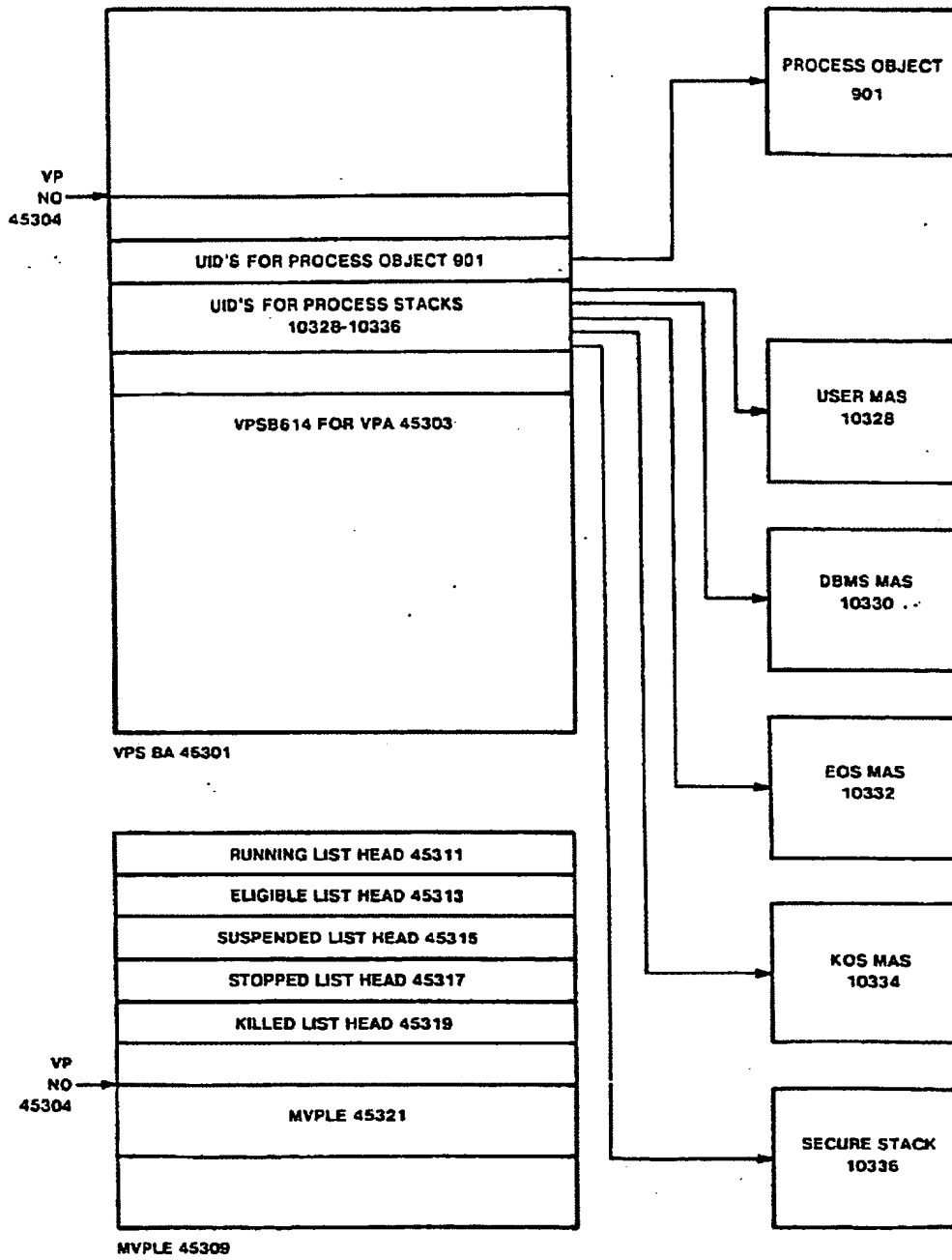


FIG. 453

VIRTUAL PROCESSOR SYNCHRONIZATION

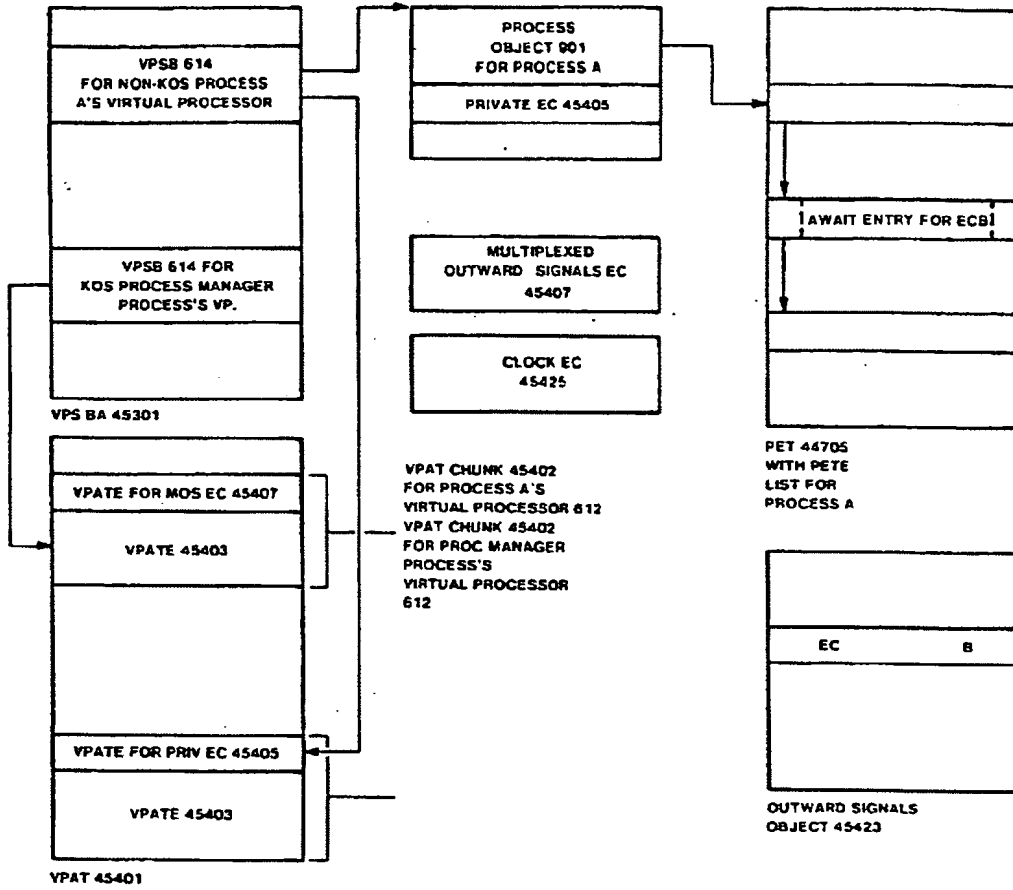


FIG. 454

MAS OBJECT OVERVIEW

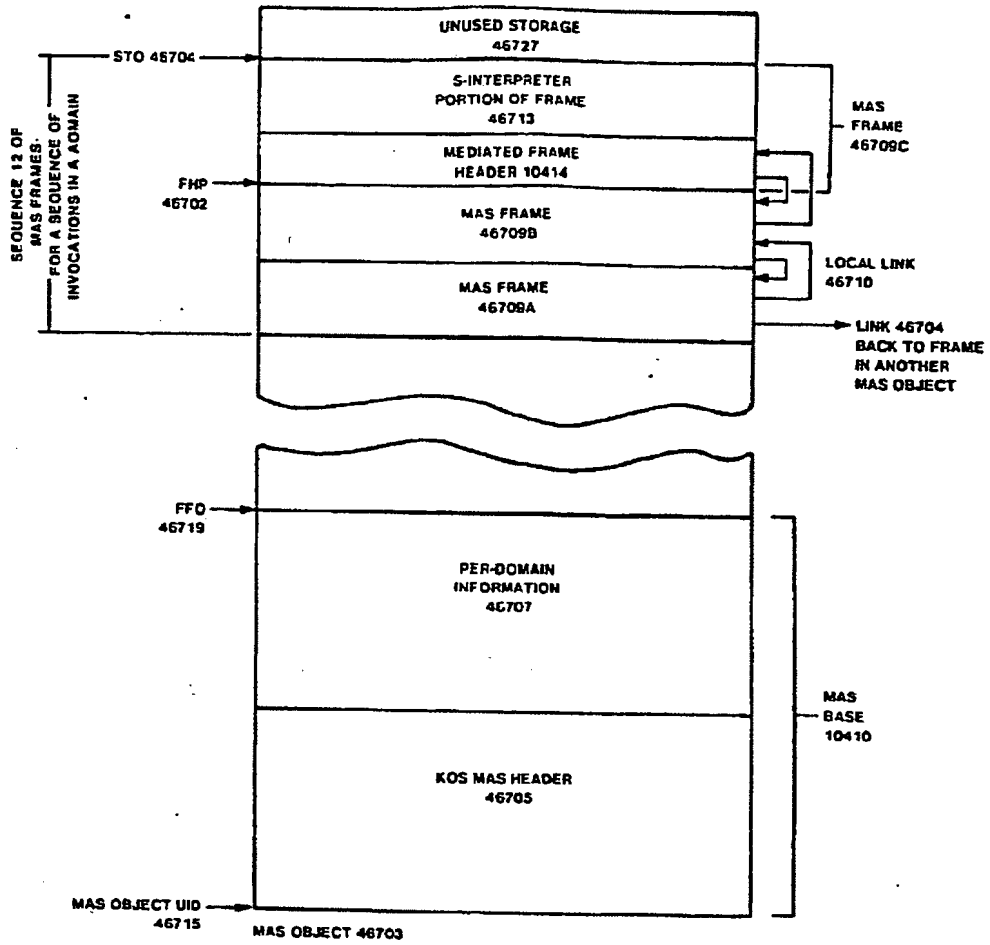


FIG. 467

MAS BOSE DETAIL

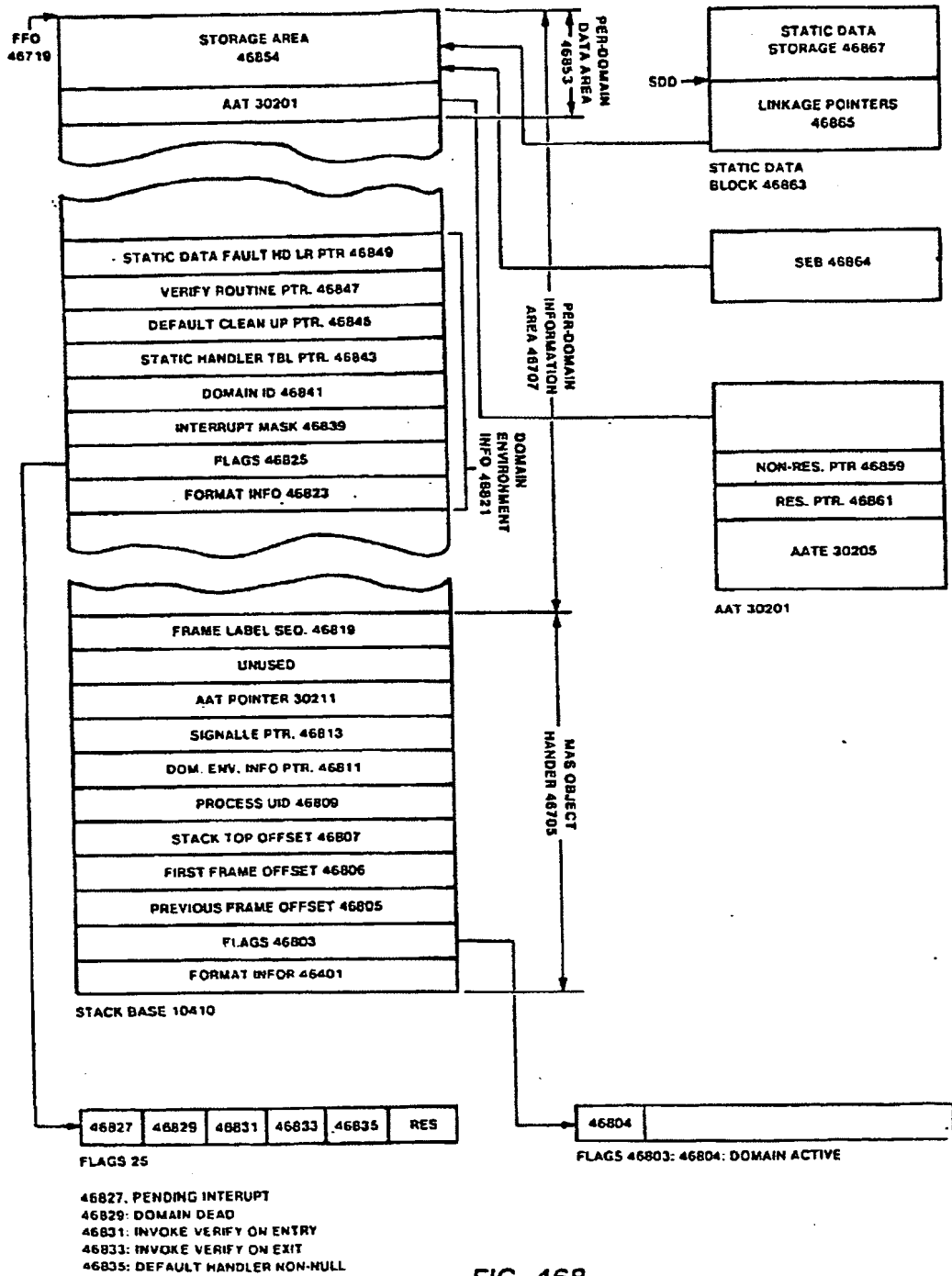
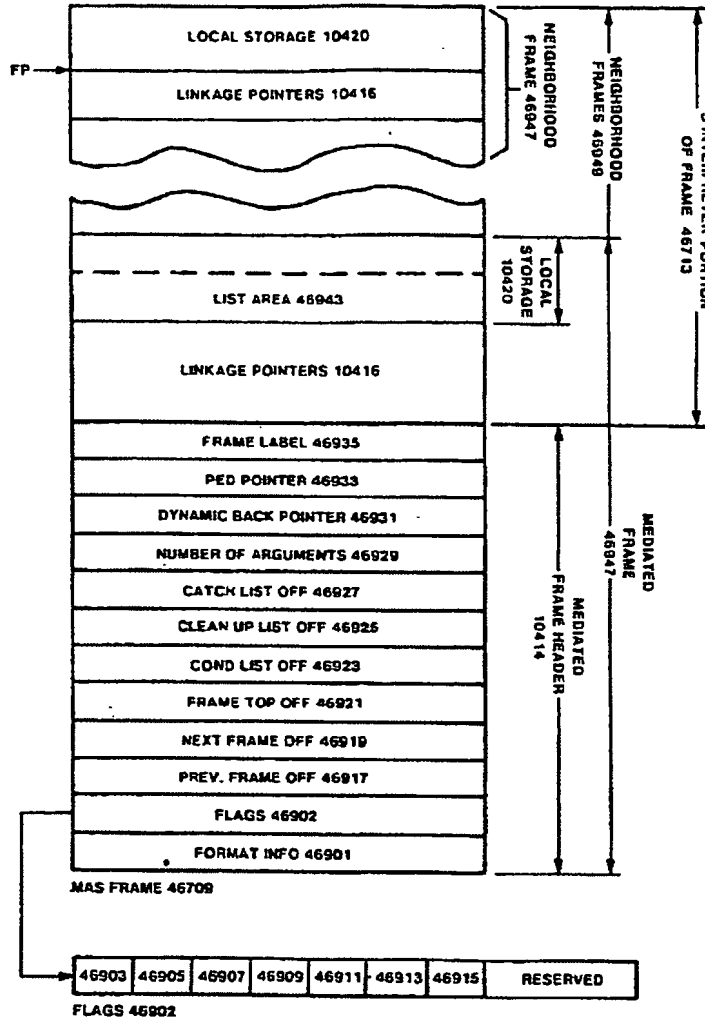


FIG. 468

MAS FRAME DETAIL



FLAGS 46902

46903: RESULT OF CROSS-DOMAIN
 46905: IN SIGNALLER
 46907: DO NOT RETURN
 46909-15: LIST PRESENT FLAGS

NOTE: IN A FRAME RESULTING FROM A CROSS-DOMAIN CALL, PFO 17=0; IN A FRAME MAKING A CROSS-DOMAIN CALL, NFO = 0

FIG. 469

SS 10336 OVERVIEW

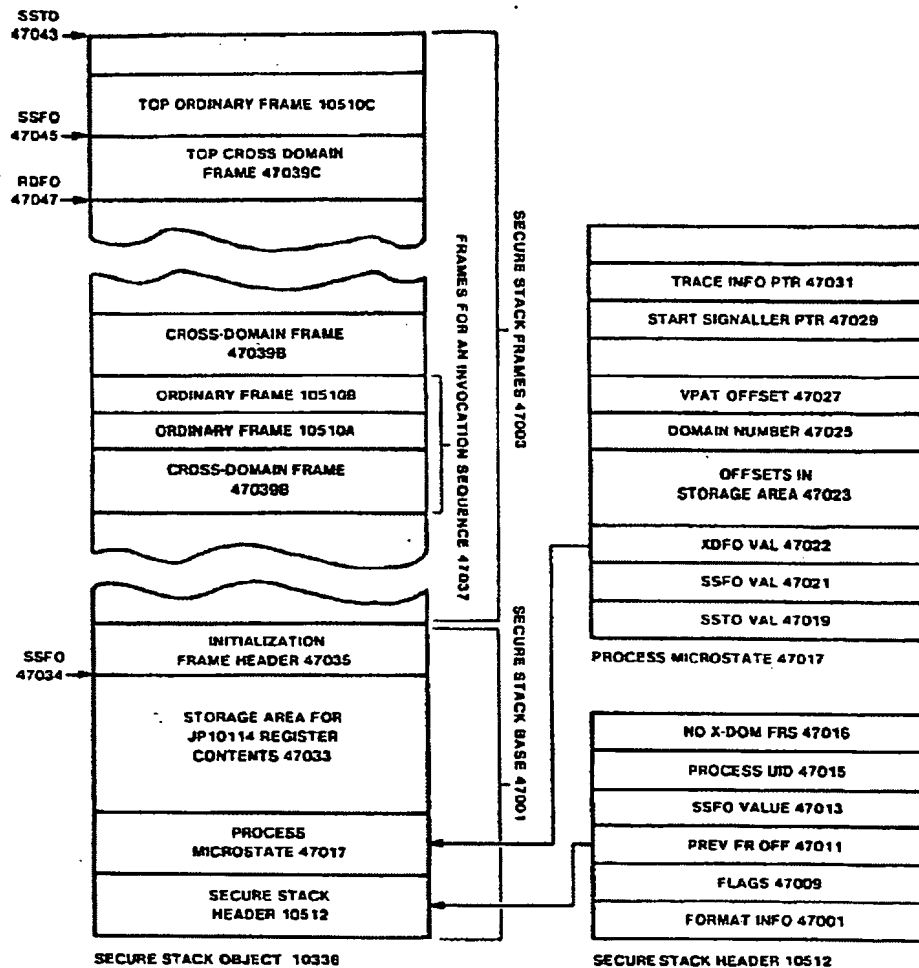


FIG. 470

SECURE STACK 10336 FRAME DETAIL

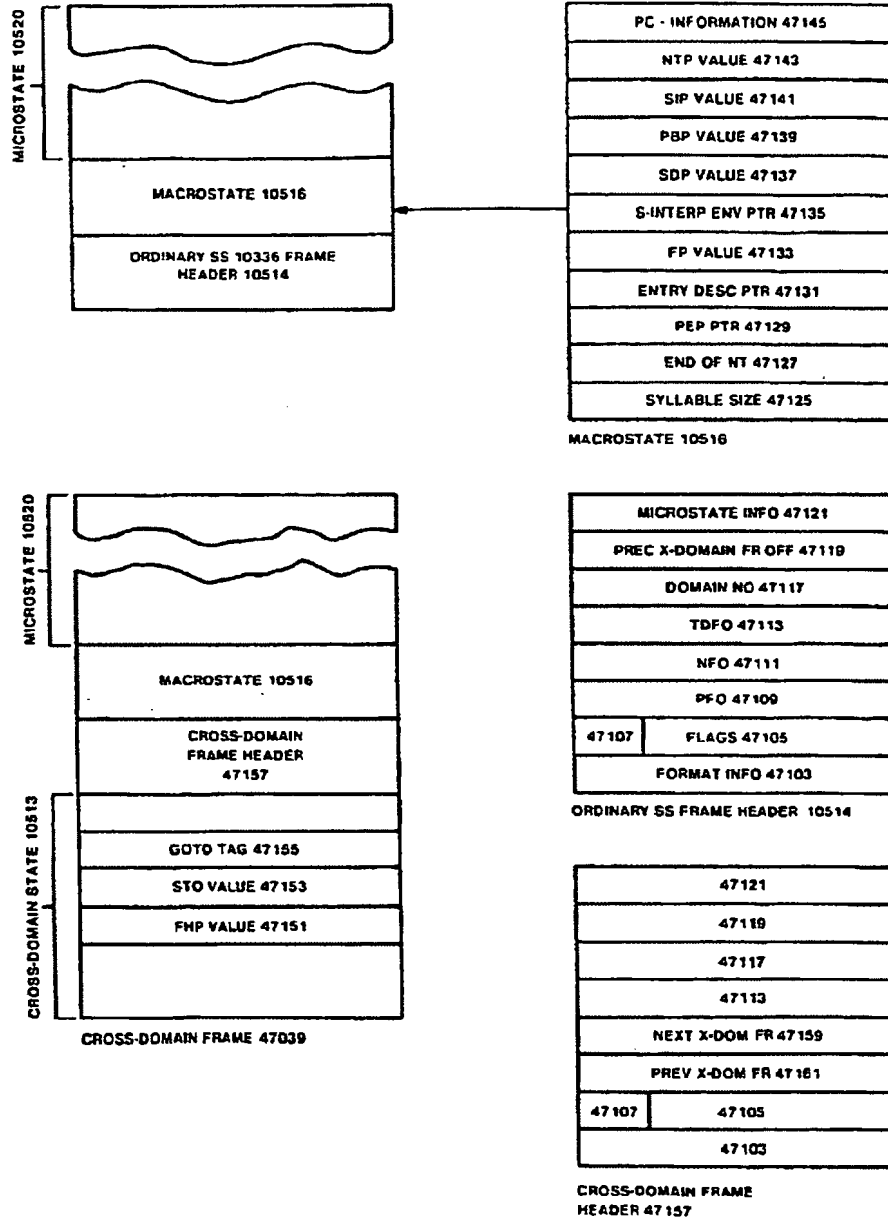
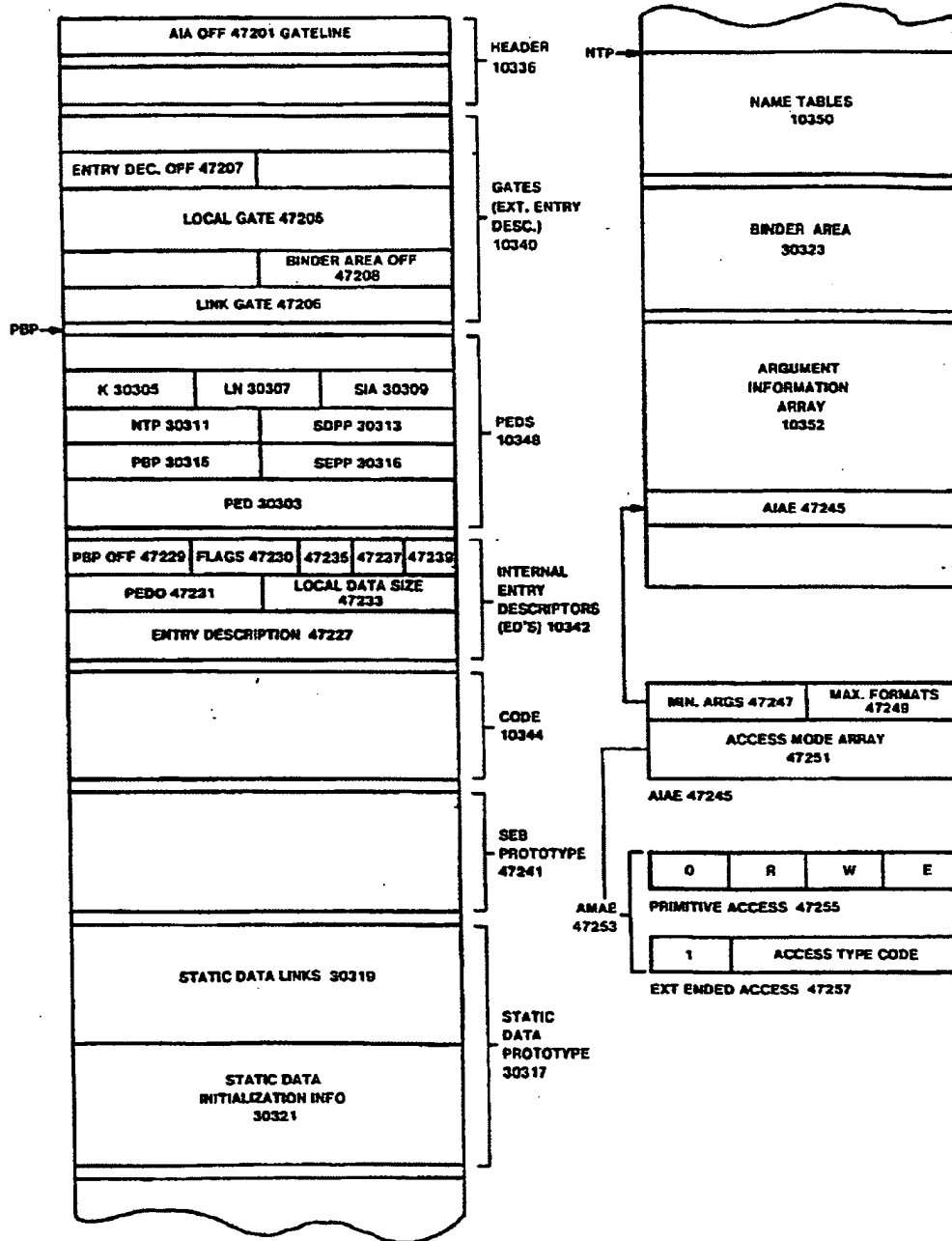


FIG. 471

PROCEDURE OBJECT OVERVIEW



47235: AIA PREVENT
 47237: SEB PRESENT
 47239: DO NOT CHECK ACCESS

FIG. 472

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
25 May 2001 (25.05.2001)

PCT

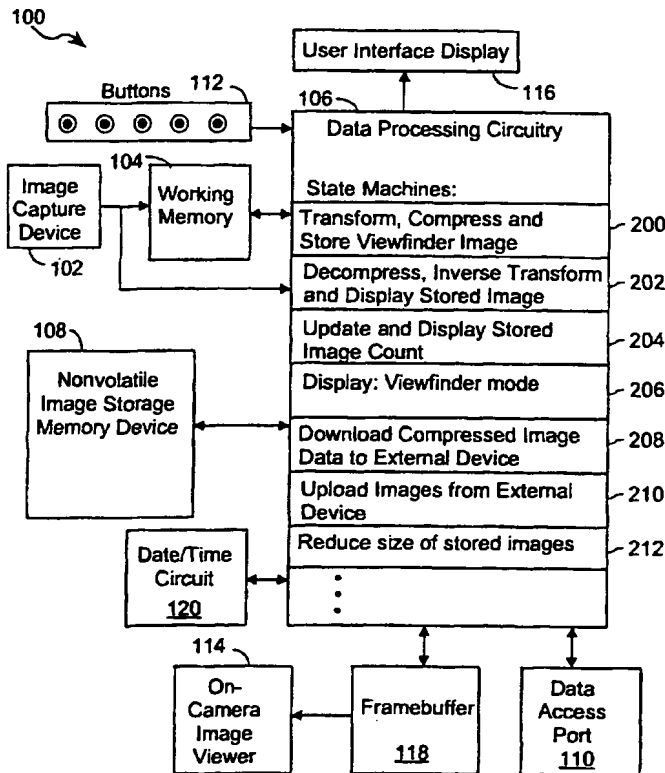
(10) International Publication Number
WO 01/37209 A1

- (51) International Patent Classification: G06K 9/36, 9/46
- (74) Agents: WILLIAMS, Gary, S. et al.; Pennie & Edmonds LLP, 1155 Avenue of the Americas, New York, NY 10036 (US).
- (21) International Application Number: PCT/US00/30825
- (81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.
- (22) International Filing Date: 8 November 2000 (08.11.2000)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data: 09/438,666 12 November 1999 (12.11.1999) US
- (84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).
- (71) Applicant: TERALOGIC, INC. [US/US]; 1240 Villa Street, Mountain View, CA 94041 (US).
- (72) Inventors: CASTOR, Jon, S.; 2160 Stockbridge Avenue, Woodside, CA 94062 (US). CHUI, Charles, K.; 340 Olive Street, Menlo Park, CA 94025 (US).

Published:
— With international search report.

[Continued on next page]

(54) Title: PICTURE AND VIDEO STORAGE MANAGEMENT SYSTEM AND METHOD



(57) Abstract: An image processing system (100) stores image files in a memory device (108) at a number of incremental quality levels. Each image file has an associated image quality (that is fidelity or resolution) level corresponding to a quality level at which the corresponding image has been encoded. The images are initially encoded by applying a predefined transform, such as a DCT transform or wavelet-like transform (200), to image data received from an image capture mechanism (102) and then applying a data compression method to the transform data (200). The image is regenerated by successively applying a data decompression method and an inverse transform to an image file (202). Image file size reduction circuitry (212) and one or more state machines are used to lower the quality level of a specified one of the image files, including circuitry for extracting a subset of the data in the specified image file and forming a lower quality version of the specified image file that occupies less space in the memory device than was previously occupied by the specified image data structure. As a result, the amount of space occupied by image files in the memory device can be reduced so as to make room for the storage of additional image files or to allow more rapid transmission in a restricted bandwidth environment.

WO 01/37209 A1



— *Before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

PICTURE AND VIDEO STORAGE MANAGEMENT SYSTEM AND METHOD

The present invention relates generally to the storage of still and video images in an image processing device or system, such as a digital camera or digital video camera or a computer based image storage system, and particularly to a system and method for storing images at different image quality levels and for enabling users to dynamically compress high quality
5 images into lower quality images in order to free up storage for the storage of additional images.

BACKGROUND OF THE INVENTION

10

Digital cameras typically include either permanent internal storage, and/or removable storage, for storing images. For instance, many digital cameras use removable flash memory cards for storing images. Each flash memory card has a fixed amount of memory, and when the memory card is full it can be removed from the camera and replaced by another flash memory
15 card. In addition, digital cameras typically have a built-in viewer that enables the user to review the images stored on the flash memory card (and/or in the internal memory) and to delete user specified ones of the stored images. Deleting stored images obviously creates room for storing additional images.

20

When a digital camera user is "in the field" he/she generally has a limited amount of image storage on hand. If all the available image storage is full, the user has the choice of either not taking any additional pictures, or of deleting pictures from the image storage devices on hand to make room for new images. While this is actually one level better than the situation with film cameras, in which the user is simply out of luck when all the available film has been used,
25 it is the premise of the present invention that the current image storage limitations of digital cameras are caused, in part, by failure to fully exploit the advantages of having images stored in digital format.

30

Similar storage vs. image quality considerations also apply to digitally encoded video frames. In particular, for any given amount of storage space, such as in a digital video camera, the goal

is to retain the best image quality for the amount of storage required for a given number of video frames. Current devices allow the user to select image quality prior to capturing a digital video image, but do not enable the user to effectively manage the storage space in the video camera with respect to video sequences already taken, other than by deletion.

5

It is an object of the present invention to provide a digital camera or digital video camera, or other constrained storage device, that can store images at a plurality of image quality (i.e., fidelity or resolution) levels and furthermore can reduce images initially stored at a first image quality level to a lower image quality level so as to reduce the amount of storage occupied by those images.

10

It is also an object of the present invention to provide space efficient and computationally efficient image and video handling mechanisms for other applications, including network connected image and video libraries, Internet web browsers executed by client computers coupled to server computers, cable television set top boxes having video storage capabilities, and so on.

15

SUMMARY OF THE INVENTION

In summary, the present invention is an image processing device or system, such as a digital camera or digital video camera or a computer based image storage system, that can store images at a number of different image quality levels. The image processing device includes a memory device and image management logic. The memory device stores image files that each represent a respective image, each image file having an associated image quality level corresponding to a quality level at which the corresponding image has been encoded. The image management logic includes data processing circuitry and state machines for storing and processing image data received from an image capture mechanism. More specifically, the image management logic includes image processing circuitry and one or more state machines for applying a predefined transform, such as a wavelet-like transform, to image data received from the image capture mechanism to generate transform image data and for applying a data compression method to the transform image data so as to generate an image file having an associated image quality level.

20

25

30

The image management logic also includes image reconstruction circuitry and one or more state machines for successively applying a data decompression method and an inverse transform to a specified one of the image files so as to generate a reconstructed image suitable for display on an image viewer.

5

Further, the image management logic includes image file size reduction circuitry and one or more state machines for reducing the size of an image file while minimizing the reduction in image quality level. This circuitry extracts a subset of the data in the specified image file and forms a lower quality version of the specified image file that occupies less space in the memory device than was previously occupied by the specified image data structure. As a result, the amount of space occupied by image files in the memory device can be reduced so as to make room for the storage of additional image files or to allow more rapid transmission in a restricted bandwidth environment.

10

In a preferred embodiment, the image transform data in an image file is organized on a bit plane basis such that image transform data for at least one bit plane is stored in distinct portions of the image data structure from image transform data for other bit planes. To generate a lower quality image file, the image size reduction circuitry extracts a portion of the image file that excludes the image transform data for at least one bit plane and replaces the image file with an image file containing the extracted portion. Further, the image data is also organized on a transform layer basis such that image transform data for at least one transform layer is stored in distinct portions of the image data structure from image transform data for other transform layers. The image size reduction circuitry can also generate a lower quality image file by extracting a portion of the image file that excludes the image transform data for at least one transform layer and replaces the image file with an image file containing the extracted portion.

20

25

BRIEF DESCRIPTION OF THE DRAWINGS

Additional objects and features of the invention will be more readily apparent from the following detailed description and appended claims when taken in conjunction with the drawings, in which:

30

Fig. 1 is a block diagram of a digital camera in accordance with an embodiment of the present invention.

5 Fig. 2 depicts an image data array divided into a set of smaller analysis arrays for purposes of encoding and data compression.

Fig. 3 schematically depicts the process of transforming a raw image data array into a transform image array and compressing the transform image array into a compressed image file.

10

Figs. 4A and 4B depict image storage data structures.

Fig. 5 is a conceptual flow chart depicting changes in the state of a digital camera as various operations are performed.

15

Fig. 6 is a conceptual flow chart depicting changes in the state of a video image sequence as the video image sequence is encoded and compressed.

20 Figs. 7, 8 and 9 show data structures used in one particular embodiment for video image sequence encoding and compression.

Fig. 10 is a conceptual diagram of an Internet server and client devices that utilize the image or and video image compressing and management features of the present invention.

25

DESCRIPTION OF THE PREFERRED EMBODIMENTS

30 In some image processing systems, an image can be stored at a number of discrete resolution levels, typically with each resolution level differing from its "neighbors" by a resolution factor of four. In other words, if the highest resolution representation (at resolution level 1) of the image contains X amount of information, the second resolution level representation contains (for example) X/4 amount of information, the third resolution level representation contains X/16 amount of information, and so on. Thus, an image's "resolution" typically means the

amount of image information, and in the context of digital image processing systems is often expressed in terms of the number of distinct pixel elements. The number of resolution levels and the particular amount of information reduction from one level to the next may vary considerably from one system to another. Further, the present invention would be equally applicable to systems having a continuous range of resolution levels.

Another concept concerning image quality is "fidelity." The fidelity of an image can be compromised even if its resolution, in terms of the number of pixels in the image, is unchanged. An image's fidelity can be reduced by reducing the number of bits used to represent the image data. For instance, if the transform coefficients used to represent an image are represented at full fidelity using 12 bits, and then the number of bits used to represent transform coefficients is reduced to 11, the fidelity of the resulting image will be reduced. In other words, the quality of the image reconstructed from the lower fidelity data will be a little less sharp than an image reconstructed from higher fidelity data.

In this document, the terms "image quality" and "quality level" will be used in the general sense of image quality, encompassing both image "resolution" and image "fidelity." Thus, a reduction in an image's "image quality" from a top image quality level to a next highest image quality level might be accomplished either by reducing the image's resolution or by reducing its fidelity, or both. In other embodiments the terms "image quality" and "quality level" may be used to refer to other aspects of image quality as well.

Digital Camera Architecture

Referring to Fig. 1, there is shown an embodiment of a digital camera system 100 in accordance with the present invention. The digital camera system 100 includes an image capture device 102, such as a CCD or CMOS sensor array or any other mechanism suitable for capturing an image as an array of digitally encoded information. Thus the image capture device is assumed to include analog to digital conversion (ADC) circuitry for converting analog image information into digital values.

A working memory 104, typically random access memory, receives digitally encoded image information from the image capture device 102. More generally, it is used to store a digitally

encoded image while the image is being transformed and compressed and otherwise processed by the camera's data processing circuitry 106. Memory 104 may be integrated on the same integrated circuit as other devices, or may be implemented using separate circuit(s).

5 The data processing circuitry 106 in one embodiment consists of hardwired logic and a set of state machines for performing a set of predefined image processing operations. In alternate embodiments the data processing circuitry 106 could be implemented in part or entirely using a fast general purpose microprocessor and a set of software procedures. However, at least using the technology available in 1999, it would be difficult to process and store full resolution
10 images (e.g., full color images having 1280 x 840 pixels) fast enough to enable the camera to be able to take, say, twenty pictures per second, which is a requirement for some commercial cameras, as well as digital video cameras. In the future, general purpose microprocessors or general purpose image data microprocessors (e.g., single instruction multiple data (SIMD) processors) may be able to provide the fast image processing needed by digital cameras, in
15 which case the data processing circuit 106 could be implemented using such a general purpose microprocessor or perhaps a hybrid processor system.

Each image, after it has been processed by the data processing circuitry 106, is typically stored as an "image file" in a nonvolatile memory storage device 108, typically implemented using
20 "flash" (i.e., EEPROM) memory technology. The nonvolatile memory storage device 108 is preferably implemented as a removable memory card. This allows the camera's user to remove one memory card, plug in another, and then take additional pictures. However, in some implementations, the nonvolatile memory storage device 108 may not be removable, in which case the camera will typically have a data access port 110 to enable the camera to
25 transfer image files to and from other devices, such as general purpose, desktop computers, computer systems and devices used to warehouse libraries of images, computer systems and devices used to store and distribute image files, and so on. Digital cameras with removable nonvolatile memory 108 may also include a data access port 110.

30 While the amount of storage in the nonvolatile image memory 108 will vary from one implementation to the next, such devices will typically have sufficient capacity to store 10 to 50 high quality images. Once the nonvolatile image memory 108 is full, if the file size reduction methodology of the present invention is not used, the only way the camera can be

used to take additional pictures is either by deleting images from the nonvolatile image memory 108 or by replacing the nonvolatile image memory 108 with another one. If neither of these options are feasible (e.g., because the user has filled all the memory cards he/she has on hand with images that he/she does not wish to delete), then no further pictures can be taken
5 until a new memory device 108 is inserted or the stored images are transferred to an external device (if available).

The digital camera 100 includes a set of buttons 112 for giving commands to the camera. In addition to the image capture button, there will typically be several other buttons to enable the
10 use to select the quality level of the next picture to be taken, to scroll through the images in memory for viewing on the camera's image viewer 114, to delete images from the nonvolatile image memory 108, and to invoke all the camera's other functions. Such other functions might include enabling the use of a flash light source, and transferring image files to and from a computer. In accordance with the present invention, the user selectable functions, selected by
15 using the buttons 112, further include reducing the size of one or more of the image files stored in the nonvolatile image memory 108 so as to make room for the storage of additional images. The buttons in one embodiment are electromechanical contact switches, but in other embodiments at least some of the buttons may be implemented as touch screen buttons on a user interface display 116, or on the image viewer 114.

20 The user interface display 116 is typically implemented either (A) as an LCD display device separate from the image viewer 114, or (B) as images displayed on the image viewer 114. Menus, user prompts, and information about the images stored in the nonvolatile image memory 108 may be displayed on the user interface display 116, regardless of how that display
25 is implemented.

After an image has been captured, processed and stored in nonvolatile image memory 108, the associated image file may be retrieved from the memory 108 for viewing on the image viewer. More specifically, the image file is converted from its transformed, compressed form back into
30 a data array suitable for storage in a framebuffer 118. The image data in the framebuffer is displayed on the image viewer 114. A date/time circuit 120 is used to keep track of the current date and time, and each stored image is typically date stamped with the date and time that the image was taken.

Image Data Structures

Referring to Fig. 2, in one embodiment the nonvolatile image memory 108 stores a directory 130 that lists all the image files 132 stored in the memory 108. Preferably, the directory 130
5 contains information for each stored image file 132, such as the date and time the image was taken, the quality level of the image and the file's location in the memory 108.

To understand the image data structure stored in each image file, it is helpful to first understand how an image file is encoded. Referring to Fig. 3, a raw image data array 140,
10 obtained from the digital camera's image capture mechanism 102 (Fig. 1), is treated as a set of non-overlapping "analysis arrays" 142 of a fixed size, such as 32 x 32, or 64 x 64 (or more generally $2^n \times 2^n$, for some integer value of n). A sufficient number of subarrays are used to cover the entire data array that is to be encoded, even if some of the subarrays overhang the edges of the data array. The overhanging portions of the subarrays are filled with zero data
15 values during the data encoding process. In a preferred embodiment, the origin of the data array is the top left corner, the first coordinate used to identify data array positions is the "Y" axis or vertical coordinate, and the second coordinate used is the "X" axis or horizontal coordinate. Thus, a position of 0,64 indicates a pixel at the top vertical position of the array, 64 pixel positions over to the right from the array origin, while a position of 32,0 indicates a
20 pixel on the left edge of the array, 32 pixel positions vertically down from the array origin.

An appropriate transform is applied to each of the analysis arrays 142, and then the resulting transform coefficients for each analysis array are quantized (e.g., divided by an appropriate value to generate integer valued, quantized transform coefficients) so as to generate a
25 transformed image array 144. In one embodiment the transform applied to the raw image data is a wavelet-like transform. In other embodiments a DCT transform could be used (which is the type of transform used in current JPEG image encoding systems), or other types of wavelet or wavelet-like transforms could be used.

30 In this document, the terms "wavelet" and "wavelet-like" are used interchangeably. Wavelet-like transforms generally have spatial frequency characteristics similar to those of conventional wavelet transforms, and are losslessly reversible, but have shorter filters that are more computationally efficient.

In one embodiment the transformed image array 144 is generated by successive applications of a wavelet-like decomposition transform. A first application of the wavelet-like decomposition transform to an initial two dimensional array of "raw" image data generates four sets of coefficients, labeled LL, HL1, LH1 and HH1. Each succeeding application of the wavelet-like decomposition transform is applied only to the LL set of coefficients generated by the previous wavelet transformation step and generates four new sets of coefficients, labeled LL, HLx, LHx and HHx, where x represents the wavelet transform "layer" or iteration. After the last wavelet-like decomposition transform iteration only one LL set remains. The total number of coefficients generated is equal to the number of data samples in the original data array. The different sets of coefficients generated by each transform iteration are sometimes called layers. The number of wavelet transform layers generated for an image is typically a function of the resolution of the initial image. Performing five to seven wavelet transformation layers is fairly typical, but more or less may be used depending on such considerations as the size of the analysis arrays, the subject matter of the image, the data processing resources available for image compression, and the like.

For the purposes of explaining the operation of the image encoding and decoding operations of the present invention, the specific type of image transform used and the specific type of data quantization used to transform a raw image file 140 into a transformed image array 142 are not relevant and therefore are not further described herein. However, a preferred embodiment of the wavelet transform and data quantization methods are described in U.S. Patent No. 5,909,518, "System and Method for Performing Wavelet and Inverse Wavelet Like Transformations of Digital Data," which is hereby incorporated by reference as background information.

Each transformed image array 144 is compressed and encoded using a sparse data encoding technique. In one embodiment, the method of compressing and encoding the analysis arrays is the method described in detail in U.S. patent application 08/858,035, filed May 16, 1997, entitled "System and Method for Scalable Coding of Sparse Data Sets," now U.S. Patent No. 5,949,911, which is hereby incorporated by reference as background information. The encoded image data for all the analysis arrays of the image are combined and stored as an image file 132.

Referring to Fig. 4A, the image file 132 includes header data 160 and a sequence of data structures 162, each representing one analysis array. The header data 160 indicates the size of the image file and the image file's quality level. The header data also includes a list of analysis array size values indicating the length of each of the analysis array data structures 162, thereby enabling fast indexing into the image data. Storing size values for the analysis arrays enables the camera's data processing circuitry 106 (Fig. 1) to locate the beginning of any analysis array data structure 162 without having to decode the contents of the earlier analysis arrays in the image file 132.

As shown in Fig. 4B, the encoded data 162 representing any one analysis array is stored in "bit layer order". For each analysis array, the encoding procedure determines the most significant non-zero bit in the data to be encoded, which is herein called the y^{th} bit. The value of y is determined by computing the maximum number of bits required to encode the absolute value of any data value in the analysis array. In particular, y is equal to $\text{int}(\log_2 V) + 1$, where V is the largest absolute value of any element in the analysis array, and "int()" represents the integer portion of a specified value.

The encoded data 162 representing one analysis array includes (A) header data 170 indicating the maximum number of bits required to encode the absolute value of any data value in the analysis array, and (B) a sequence of data structures 172, each representing one bit plane of the elements in the analysis array. The x^{th} bit plane of the analysis array is the x^{th} bit of the absolute value of each of the elements in the analysis array. A sparse data encoding technique is used so that it takes very little data to represent a bit plane that contains mostly zero values. Typically, higher frequency portions of the transformed, quantized image data will contain more zero values than non-zero values, and further most of the non-zero values will have relatively small absolute value. Therefore, the higher level bit planes of many analysis arrays will be populated with very few non-zero bit values.

In an alternate embodiment, the data structure shown in Fig. 4A is modified slightly. In particular, to facilitate fast extraction of lower-resolution image data from an image file, the boundaries of the analysis arrays are adjusted, if necessary, so as to coincide precisely with the boundaries between the wavelet transform regions shown in Fig. 3 (e.g., the boundary between HL2 and HL1). If the size of the initial image array is not equal to an integer number of

analysis arrays (i.e., if either the height or width of the image array is not an integer multiple of 2^n , where the size of each analysis array is $2^n \times 2^n$ for an integer value of n), at least some of the boundaries between wavelet transform regions will fall in the middle of the analysis regions. For example, for a 800 x 600 pixel image, the LL region might have a size of 50 x 38. If the
5 wavelet transform coefficients are encoded in units of analysis regions of size 32 x 32, the LL region will be encoded in four analysis regions, three of which would normally contain data for neighboring wavelet transform regions. In this alternate embodiment, each analysis array that overlaps a border between wavelet transform regions is replaced by two or four analysis regions (depending on whether the analysis array overlaps one or two region boundaries), with
10 zero values being stored in the appropriate locations so that each analysis array contains data from only one wavelet transform region. The analysis arrays are still stored in "origin sorted order" in the image file 132, with the "origin" now being defined as the coordinate of the coefficient closest to the upper left corner of the analysis array that has not been overwritten with zero values.

15 In another alternate embodiment, a different transform than the wavelet-like transform could be used, but the resulting image data would still be stored in bit plane order. For instance, a DCT transform could be used.

20 In some embodiments of the present invention, the raw image array received from the digital camera's image capture mechanism may first be divided into "analysis arrays" and then transformed and quantized. Further, the analysis arrays may each be a thin horizontal strip of the image array. That is, each analysis array may extend the full width of the image array, but have a height of only a few (e.g., 4 to 16) image elements. In yet another embodiment, the
25 image array might not be divided into analysis arrays at all.

Generally, in all embodiments described above, the compressed encoded image data is stored in bit plane order. The reason that bit plane ordered storage is favored is that it makes gradual fidelity reduction very easy: to reduce the fidelity of an image file by a minimum amount, the
30 data for the lowest level bit plane in the file is discarded and the remaining image data is retained, resulting in a smaller file with one bit plane less fidelity.

However, in yet other alternate embodiments, each image file could be organized in "quality level support order" with the data for each analysis array being arranged so that each successive data structure 172 stores the image information needed to increase image quality by one predefined image quality level. Thus, the information in some data structures 172 might represent two, three or more bit planes of information. In this embodiment, an image can be reduced by one quality level by deleting the last data structure 172 from every analysis array data structure 172 in the image file.

Digital Camera State Machines

For the purposes of this explanation, it will be assumed that the digital camera 100 has four predefined image quality levels: High, Very Good +, Very Good -, and Good. It will be further assumed that image files stored at High quality typically occupy about twice as much space as image files stored at Good quality. In other embodiments, more or fewer image quality levels could be used, and the ratio of image file sizes from highest to lowest quality could be larger or smaller than 2:1. For instance, if the camera is capable of taking very high resolution images, such as 2000 x 2000 pixels or even 4000 x 4000 pixels, and at very high fidelity, then it would make sense to provide a large number of quality levels with a ratio of image file sizes from highest to lowest quality of perhaps as high as 64:1.

It is noted that an image file's quality cannot be increased once it has been lowered, unless the original image file or an alternate source thereof remains available, because the information needed to restore the image's quality has been lost.

Referring back to Fig. 1, the digital camera 100 preferably includes data processing circuitry 106 for performing a predefined set of primitive operations, such as performing the multiply and addition operations required to apply a transform to a certain amount of image data, as well as a set of state machines 200-212 for controlling the data processing circuitry so as to perform a set of predefined image handling operations. In one embodiment, the state machines in the digital camera are as follows.

- One or more state machines 200 for transforming, compressing and storing an image received from the camera's image capture mechanism. This image is sometimes called the "viewfinder" image, since the image being processed is generally the one seen on the camera's

image viewer 114. This set of state machines 200 are the ones that initially generate each image file stored in the nonvolatile image memory 108. Prior to taking the picture, the user specifies the quality level of the image to be stored, using the camera's buttons 112. It should be noted that in most digital cameras the viewfinder is capable of displaying only a very small and low fidelity version of the captured image, and thus the image displayed in the camera's viewfinder is typically a much lower quality rendition of the captured image than the quality of the "viewfinder" image stored in the image file.

• One or more state machines 202 for decompressing, inverse transforming and displaying a stored image file on the camera's image viewer. The reconstructed image generated by decompressing, inverse transforming and dequantizing the image data is stored in camera's framebuffer 118 so that it can be viewed on the image viewer 114.

• One or more state machines 204 for updating and displaying a count of the number of images stored in the nonvolatile image memory 108. The image count is preferably displayed on the user interface display 116. This set of state machines 204 will also typically indicate what percentage of the nonvolatile image memory 108 remains unoccupied by image files, or some other indication of the camera's ability to store additional images. If the camera does not have a separate interface display 116, this memory status information may be shown on the image viewer 114, for instance superimposed on the image shown in the image viewer 114 or shown in a region of the viewer 114 separate from the main viewer image.

• One or more state machines 206 for implementing a "viewfinder" mode for the camera in which the image currently "seen" by the image capture mechanism 102 is displayed on the image viewer 114 to that the user can see the image that would be stored if the image capture button is pressed. These state machines transfer the image received from the image capture device 102, possibly after appropriate remedial processing steps are performed to improve the raw image data, to the camera's framebuffer 118.

• One or more state machines 208 for downloading images from the nonvolatile image memory 108 to an external device, such as a general purpose computer.

• One or more state machines 210 for uploading images from an external device, such as a general purpose computer, into the nonvolatile image memory 108. This enables the camera to be used as an image viewing device, and also as a mechanism for transferring image files on memory cards.

• One or more state machines 212 for reducing the size of image files in the nonvolatile image memory 108. This will be described in more detail next.

In the context of the present invention, an image file's quality level can be reduced in one of two ways: 1) by deleting from the image file all the analysis arrays associated with one or more transform layers, or 2) by deleting from the image file one or more bit planes of data. In either method, the state machines 212 extract the data structures of the image file that correspond to the new, lower image quality level selected by the user, and then replaces the original image file with one that stores the extracted data structures. Alternately, the original image file is updated by deleting a portion of its contents, thereby freeing some of the memory previously occupied by the file. A feature of the present invention is that the image quality level of an image file can be lowered without having to reconstruct the image and then re-encode it, which would be costly in terms of the computational resources used. Rather, the data structures within the image file are pre-arranged so that the image data in the file does not need to be read, analyzed or reconstructed. The image quality level of an image file is lowered simply by keeping an easily determined subset of the data in the image file and deleting the remainder of the data in the image file, or equivalently by extracting and storing in a new image file a determined subset of the data in the image file and deleting the original image file.

For the purposes of this document, it should be noted that the term "deleting" when applied to a data structure in an image file does not necessarily mean that the information in the data structure is replaced with null values. Rather, what this means is that the image file is replaced with another image file that does not contain the "deleted" data structure. Thus, various data structures in an image file may be deleted simply by copying all the other data structures in the image file into a new image file, and updating all required bookkeeping information in the image file and the image directory for the modified file. The "deleted" data structures may actually remain in memory unchanged until they are overwritten with new information. Alternately, data in an image file may in some implementations be deleted solely by updating the bookkeeping information for the file, without moving any of the image data.

In one embodiment of the present invention, the digital camera lowers the image quality of an image from High quality to "Very Good +" by deleting the two lowest bit planes of the image. Similarly, lowering the image's quality to "Very Good -" is accomplished by deleting two more bit planes of the image, and then lowering the image's quality to Good is accomplished by deleting yet another two bit planes of the image. More generally, each quality level

transition is represented by deleting a certain percentage of the bit planes of the highest quality level representation of the image.

5 In an alternate embodiment, the transition from High quality to the next highest quality level is accomplished by deleting the analysis arrays for a first transform layer (e.g., the analysis arrays for the HL1, HH1 and LH1 regions of the transformed image in Fig. 3). Subsequent quality level transitions to lower quality levels are accomplished by deleting appropriate numbers of bit planes.

10 In one embodiment of the present invention the digital camera provides two image file size reduction modes. In a first size reduction mode, the user selects one image (or a specified group of images), uses the camera's buttons to indicate what lower quality level the image is to be stored at, and then the state machine 212 generates a smaller image file and stores the resulting image file in the camera's nonvolatile image memory 108. In the second size
15 reduction mode, the user commands the camera to reduce the size of all image files that are currently stored at quality level A to quality level B. For instance, in this second size reduction mode the user might command the camera to convert all "High" quality image files to "Very Good +" image quality files. This latter size reduction mode is particularly useful for "clearing space" in memory 108 to enable additional pictures to be stored in the memory 108.

20 In another embodiment, the camera or other device may include one or more automatic image file size reduction modes. For instance, in one such mode the camera could be set to record all pictures at a particular quality level. When the camera's memory is sufficiently filed with images files so that there is insufficient room to store one more image at the current quality
25 level setting, the camera automatically reduces the size of enough of the stored image files so as to create room for one more image at the current quality level. In some embodiments, the quality level setting of the device for future images might be automatically reduced to match the quality level of the highest quality image stored in the camera's memory. In this way, the camera takes and stores the maximum quality images for the space available, and this
30 maximization will occur flexibly and "on-the-fly."

Camera Operation and
Image File Size Reduction

Referring to Fig. 5, the status of a digital camera is represented by status information displayed
5 on the camera's user interface display 116. For example, before the camera's image memory
108 is filled, the camera might indicate to the user that it is currently storing twenty-one
pictures at High quality and has enough memory to store three more pictures. The indication
of how many more pictures can be stored in the camera's image memory 108 (Fig. 1) depends
10 on the camera's current picture quality setting, which determines the quality of the next picture
to be taken.

After the camera has stored three more pictures, the camera's image memory 108 is full (i.e., it
has insufficient room to store another picture at the camera's current picture quality setting),
and the camera indicates to the user that it is currently storing twenty-four pictures at High
15 quality and has enough memory to store zero more pictures. For the purposes of this example,
we will assume that the user wants to take at least ten more pictures, despite the fact that he/she
has no more memory cards. To make this possible, the user utilizes the image size reduction
feature of the camera.

20 In this example, the user commands the camera to reduce all "High" quality image files down
one quality level to the "Very Good +" quality level. The camera accomplishes this by running
the size reduction state machine 212 and then updating the status information displayed on the
camera's user interface display 116. In this example, the twenty-four images are now shown to
be stored in image files having the "Very Good +" quality level, and the camera has room for
25 seven new images at the High quality image level.

In this example, the user next commands the camera to perform a second size reduction so as
to compress all "Very Good +" quality image files down one quality level to the "Very
Good -" quality level. The camera accomplishes this by running the size reduction state
30 machine 212 and then updating the status information displayed on the camera's user interface
display 116. In this example, the twenty-four images are now shown to be stored in image
files having the "Very Good -" quality level, and the camera has room for twelve new images
at the High quality image level.

Alternately, if the user had, before capturing the images, switched the quality level for new images to "Very Good +" quality, a single image size reduction step might have been sufficient to create room for at least ten additional pictures.

5 In another example, the digital camera may be configured to have an automatic image file size reduction mode that is activated only when the camera's memory is full and the user nevertheless presses the image capture button on the camera. In this mode of operation, the camera's image processing circuitry reduces the size of previously stored image files as little as possible so as to make room for an additional image file. If the user continues to take more
10 pictures in this mode, the quality of the stored images will eventually degrade to some user defined or predefined setting for the lowest allowed quality level, at which point the camera will not store any additional image files until the permitted quality level is lowered further or at least some of the previously stored image files are transferred to another device or otherwise deleted.

15 The image management system and method of the present invention can also be implemented in computer systems and computer controlled systems, using a data processor that executes procedures for carrying of the image processing steps discussed above. The present invention can also be implemented as a computer program product (e.g., a CD-ROM or data signal
20 conveyed on a carrier signal) containing image processing procedures suitable for use by a computer system.

Video Image Management System

25 Referring to Fig. 6, there is shown a conceptual data flow diagram for a video image management system for storing video images at a plurality of image quality levels. The basic structure of the video image management system is the same as shown in Fig. 1. However, when the camera is a digital video camera, successive images F_i are automatically generated at a predefined rate, such as eight, sixteen, twenty-four or thirty frames per second. In a preferred
30 embodiment, the sequence of video images is processed N frames at a time, where N is an integer greater than three, and is preferably equal to four, eight or sixteen; generally N will be determined by the availability of memory and computational resources in the particular system in which the invention is being implemented. That is, each set of N (e.g., sixteen) successive

images (i.e., frames) are processed as a set, as follows. For each set of sixteen frames F_{16n} to F_{16n+15} , all the frames except the first one are replaced with differential frames. Thus, when $N=16$, fifteen differential frames $F_{i+1}-F_i$ are generated. Then, following the data processing method shown in Fig. 3 and discussed above, the first frame and the fifteen differential frames
5 are each divided into analysis arrays, a wavelet-like or other transform is applied to the analysis arrays, and then the resulting transform coefficients are encoded.

In alternate embodiments, other methodologies could be used for initially transforming and encoding each set of success frames. For instance, a frame by frame decision might be made,
10 based on a measurement of frame similarity or dissimilarity, as to whether or not to replace the frame with a differential frame before applying the transform and encoding steps.

In all embodiments, the image file (or files) representing the set of frames is stored so as to facilitate the generation of smaller image files with minimal computational resources. In particular, the data in the image file(s) is preferably stored using distinct data structures for each bit plane (see Fig. 4B). Furthermore, as explained above with reference to Fig. 4A, the analysis arrays may be adjusted prior to the transform step so that the boundaries between analysis arrays correspond to the boundaries between transform layer coefficients. By so
15 arranging the data stored in the video image files, the generation of smaller, lower quality level video image files is made much easier.
20

Continuing to refer to Fig. 6, after the video image files for a video frame sequence have been generated at a particular initial quality level, the user of the device (or the device operating in a particular automatic mode) may decide to reduce the size of the video image files while
25 retaining as much image quality as possible. By way of example, in a first video image file size reduction step, the HH1 transform coefficients for the last eight frames of each sixteen frame sequence are deleted. In a second reduction step, the HH1 transform coefficients are deleted for all frames other than the first frame of each sixteen frame sequence. In a third reduction step, the Z (e.g., four) least significant bit planes of the video image files are deleted.
30 In a fourth reduction step, the HL1 and LH1 coefficients are deleted for the last eight frames of each sixteen frame sequence. In a fifth reduction step, the HL1 and LH1 coefficients are deleted for all frames other than the first frame of each sixteen frame sequence. These reduction steps are only examples of the type of file size reduction steps that could be

performed. For instance, in other embodiments, bit planes might be deleted in earlier reduction steps and transform layers (or portions of transform layers) deleted only in later reduction steps. In general, each video image size reduction causes a corresponding decrease in image quality.

5

Referring to Figs. 7, 8 and 9, in another embodiment, video image sequences are compressed and encoded by performing a time domain, one dimensional wavelet transformation on a set of video frames. In particular, the video frames are divided into groups of N frames, where N is an integer greater than 3, and for every x,y pixel position in the video image, a one dimensional

10 K level wavelet transform is performed on the pixels for a sequence of N+1 frames. For instance, the K level wavelet transform is performed on the 1,1 pixels for the last frame of the previous group and the current group of N frames, as well as the 1, 2 pixels, the 1,3 pixels and so on.

15

In order to avoid artifacts from the separate encoding of each group of N frames, the frame immediately preceding the current group is included in K level wavelet transform. The wavelet transform uses a short transform filter that extends only one position to the left (backwards in time) of the position for which a coefficient is being generated and extends in the right hand (forward in time) direction only to the right hand edge of the set of N frames.

20

Furthermore, as shown in Fig. 8, the "right edge" coefficients are saved for each of the first through K-1 level wavelet transforms for use when processing the next group of N frames. In a preferred embodiment, only the rightmost edge coefficient is saved for each of the first through K-1 level transforms; in other embodiments two or more right edge coefficients may

25 be saved and used when processing the next block of N frames. When the second level transform is performed on a block of N frames, the saved layer 1 right edge coefficients for the previous set of N frames are used (i.e., included in the computation of the leftmost computed coefficient(s) for layer 2). By saving the rightmost edge coefficients for each of the 1 through

30 K-1 layers, artifacts that would caused (during regeneration of the video image sequence) by the discontinuities between the last frame of one block and the first frame of the next block are avoided, resulting in a smoother and more visually pleasing reconstructed video image sequence. The wavelet-like transformation and data compression of a video sequence is shown in pseudocode form in Table 1.

Table 1
Pseudocode for Wavelet-Like Transform and
Compression of One Block of Video Frames

```

5  Repeat for each block of video frames:
    {
    For each row y (of the images)
10      {
        For each column x (of the images)
            {
                Save rightmost edge value for use when processing next block of video frames;

                Apply level 1 wavelet-like transform to time-ordered sequence of pixel values
15                at position x,y, including saved edge value from prior block to generate level 1
                L and H coefficients;

                Save rightmost edge L coefficient for use when processing next block of video
                frames;
20                Apply level 2 wavelet-like transform to level 1 L coefficients for position x,y,
                including saved level 1 edge value from prior block to generate level 2 L and H
                coefficients;

25                Save rightmost edge level 2 L coefficient for use when processing next block of
                video frames;

                ...

30                Apply level k-1 wavelet-like transform to level k-2 L coefficients for position
                x,y, including saved level k-2 edge value from prior block to generate level k-1
                L and H coefficients;

                Save rightmost edge level k-1 L coefficient for use when processing next block
35                of video frames;

                Apply level k wavelet-like transform to level k-1 L coefficients for position x,y,
                including saved level k-1 edge value from prior block to generate level k L and
                H coefficients;
40            }
        }
    }
    Quantize coefficients
    Encode coefficients
    Store coefficients in image data structure(s), creating image file for current block of video
45    frames
    }

```

A more detailed explanation of saving edge coefficient from one block of image data for use while performing a wavelet or wavelet like transforms on a neighboring block of image data is provided in U.S. patent application serial no. 09/358,876, filed 07-22-99, "Memory Saving Wavelet-Like Image Transform System and Method for Digital Camera And Other Memory Conservative Applications," which is hereby incorporated by reference as background information.

Once the wavelet-like transform of each block of video data has been completed, all other aspects of processing the transformed video data are as described above. That is, the transformed data is quantized, stored in image data structures and subject to reductions in image quality, using the same techniques as those applied to still images and video image sequences as described above.

The video image management system and method of the present invention can also be implemented in computer systems and computer controlled systems, using a data processor that executes procedures for carrying of the video frame processing steps discussed above. The present invention can also be implemented as a computer program product (e.g., a CD-ROM or data signal conveyed on a carrier signal) containing image and/or video frame processing procedures suitable for use by a computer system.

Alternate Embodiments

The state machines of the embodiments described above can be replaced by software procedures that are executed by a general purpose (programmable) data processor or a programmable image data processor, especially if speed of operation is not a concern.

Numerous other aspects of the described embodiments may change over time as technology improvements are used to upgrade various parts of the digital camera. For instance, the memory technology used to store image files might change from flash memory to another type of memory, or a camera might respond to voice commands, enabling the use of fewer buttons.

Referring to Fig. 10, the present invention can also be used in a variety of image processing systems other than digital cameras and digital video cameras, including cable television set top

boxes, computer systems and devices used to warehouse libraries of images, computer systems and devices used to store and distribute image files, and so on. For example, an Internet server 300 can store images and/or video sequences in the wavelet transform compressed data structures of the present invention. Copies of those compressed data structures are transferred

5 to the memory 306 of client computers or other client devices 302, using HTTP or any other suitable protocol via the Internet 304 or other communications network. When appropriate, an image or video sequence is reduced in size so as to fit in the memory available in the client computer or other device (client device). Furthermore, once an image or video sequence has been stored in the memory 306 of a client device, the techniques of the present invention can

10 be used to manage the storage of the image, for instance through gradual reduction of image quality so as to make room for the storage of additional images or video sequences. In the embodiment shown in Fig. 10, the memory 306 of the client computer will have stored therein:

- an operating system 310;
- a browser or other image viewer application 312 for viewing documents and images;

15

- image files 314;
- image transform procedures 316, such as wavelet or wavelet-like transform procedures for converting a raw image array into wavelet transform coefficients, procedures for compressing and encoding the wavelet transform coefficients, as well as other transform procedures for handling images received in other image formats, such JPEG transform

20

- procedures for converting JPEG files into reconstructed image data that is then used as the raw image data by a wavelet or wavelet-like transform procedure;
- an image compression, quality reduction procedure 318 for implementing the image data structure size and quality reduction features of the present invention; and
- image reconstruction procedures 320 for decompressing and reverse transforming

25

- image files so as to generate reconstructed image data arrays that are suitable for viewing on the monitor of the client workstation, or for printing or other use.

The client workstation memory 306 will typically include both high speed random access memory and slower non-volatile memory such as a hard disk and/or read-only memory. The

30

client workstation's central processing unit(s) 308 execute operating system procedures and image handling procedures, as well as other applications, thereby performing image processing functions similar to those performed by dedicated circuitry in other embodiments of the present invention.

As indicated above, when the present invention is used in conjunction with, or as part of, a browser application, for management of image storage, some images may be initially received in formats other than "raw" image arrays. For instance, some images may be initially received as JPEG files, or in other proprietary or industry standard formats. To make full use of the capabilities of the present invention, such images are preferably decoded so as to generate reconstructed "raw" image arrays, and then those raw image arrays are wavelet or wavelet-like transformed so as to put the images in a form that enables use of the image quality level management features of the present invention.

10 While the present invention has been described with reference to a few specific embodiments, the description is illustrative of the invention and is not to be construed as limiting the invention. Various modifications may occur to those skilled in the art without departing from the true spirit and scope of the invention as defined by the appended claims.

WHAT IS CLAIMED IS:

- 1 1. Image processing apparatus, for use in conjunction with an image capture mechanism,
2 the image processing apparatus comprising:
3 a memory device for storing a plurality of image data structures that each represent a
4 respective image, each image data structure having an associated image quality level
5 corresponding to a quality level at which the corresponding image has been encoded in the
6 image data structure; the image quality level of each image data structure being a member of
7 predefined range of image quality levels that range from a highest quality level to a lowest
8 quality level and that include at least two distinct quality levels;
9 image management logic, including data processing circuitry and state machines for
10 storing and processing image data received from the image capture mechanism, the data
11 processing circuitry and state machines including:
12 image processing circuitry for applying a predefined transform to image data
13 received from the image capture mechanism to generate transform image data and for applying
14 a data compression method to the transform image data so as to generate a new image data
15 structure having an associated image quality level selected from the predefined range of image
16 quality levels; the new image data structure being stored in the memory device;
17 image size reduction circuitry for extracting a subset of the data in a first
18 specified one of the image data structures stored in the memory device, and forming a lower
19 quality version of the first specified image data structure that occupies less space in the
20 memory device than was previously occupied by the first specified image data structure; and
21 image reconstruction circuitry for successively applying a data decompression
22 method and an inverse transform to any specified one of the image data structures so as to
23 generate a reconstructed image suitable for display on an image viewer;
24 wherein the amount of space occupied by images stored in the form of image data
25 structures in the memory device can be reduced so as to make room for the storage of
26 additional image data structures in the memory device.
- 1 2. The image processing apparatus of claim 1, wherein
2 each image data structure contains image transform data organized on a bit plane basis
3 such that image transform data for at least one bit plane is stored in distinct portions of the
4 image data structure from image transform data for other bit planes; and

5 the image size reduction circuitry and one or more state machines includes logic for
6 extracting a portion of an image data structure that excludes the image transform data for at
7 least one bit plane and for replacing the image data structure with an image data structure
8 containing the extracted portion.

1 3. The image processing apparatus of claim 1, wherein
2 each of a subset of the image data structures contains image transform data organized
3 on a transform layer basis such that image transform data for at least one transform layer is
4 stored in distinct portions of the image data structure from image transform data for other
5 transform layers; and
6 the image size reduction circuitry and one or more state machines includes logic,
7 operative when the first specified data structure is a member of the subset of image data
8 structures, for extracting a portion of the first specified image data structure that excludes the
9 image transform data for at least one transform layer and for replacing the first specified image
10 data structure with an image data structure containing the extracted portion.

1 4. Image processing apparatus, for use in conjunction with an image capture mechanism,
2 the image processing apparatus comprising:
3 a memory device for storing a plurality of image data structures that each represent a
4 respective image, each image data structure having an associated image quality level
5 corresponding to a quality level at which the corresponding image has been encoded in the
6 image data structure; the image quality level of each image data structure being a member of
7 predefined range of image quality levels that range from a highest quality level to a lowest
8 quality level and that include at least two distinct quality levels;
9 image management logic for storing and processing image data received from the
10 image capture mechanism, including:
11 a data processor coupled to the memory device;
12 image management procedures, executable by the data processor, including instructions
13 for storing and processing image data received from the image capture mechanism, the
14 instructions including:
15 an initial image processing procedure for applying a predefined transform to
16 image data received from the image capture mechanism to generate transform image data and
17 for applying a data compression procedure to the transform image data so as to generate an

18 image data structure having an associated image quality level selected from the predefined
19 range of image quality levels;
20 an image size reduction procedure for lowering the quality level of a first
21 specified one of the image data structures, including instructions for extracting a subset of the
22 data in the first specified image data structure and forming a lower quality version of the first
23 specified image data structure that occupies less space in the memory device than was
24 previously occupied by the first specified image data structure; and
25 at least one image reconstruction procedure for successively applying a data
26 decompression method and an inverse transform to any specified one of the image data
27 structures stored in the memory device so as to generate a reconstructed image suitable for
28 display on an image viewer;
29 wherein the amount of space occupied by images stored in the form of image data
30 structures in the memory device can be reduced so as to make room for the storage of
31 additional image data structures in the memory device.

1 5. The image processing apparatus of claim 4, wherein
2 each of the image data structures contains image transform data organized on a bit
3 plane basis such that image transform data for at least one bit plane is stored in distinct
4 portions of the image data structure from image transform data for other bit planes; and
5 the image size reduction instructions include instructions for extracting a portion of an
6 image data structure that excludes the image transform data for at least one bit plane and for
7 replacing the image data structure with an image data structure containing the extracted
8 portion.

1 6. The image processing apparatus of claim 4, wherein
2 each of a subset of the image data structures contains image transform data organized
3 on a transform layer basis such that image transform data for at least one transform layer is
4 stored in distinct portions of the image data structure from image transform data for other
5 transform layers; and
6 the image size reduction instructions include instructions, operative when the first
7 specified data structure is a member of the subset of image data structures, for extracting a
8 portion of the first specified image data structure that excludes the image transform data for at

9 least one transform layer and for replacing the first specified image data structure with an
10 image data structure containing the extracted portion.

1 7. Image processing apparatus, comprising:
2 image management logic, including:
3 image processing circuitry for applying a predefined transform to an array of
4 image data so as to generate transform image data and for applying a data compression method
5 to the transform image data so as to generate an image data structure having an associated
6 image quality level selected from a predefined range of image quality levels that range from a
7 highest quality level to a lowest quality level and that include at least two distinct quality
8 levels;
9 a memory device for storing the image data structure and other image data structures
10 representing a set of images;
11 the image management logic further including:
12 image size reduction circuitry for extracting a subset of the data in a first
13 specified one of the image data structures stored in the memory device, and forming a reduced
14 size version of the first specified image data structure that occupies less space in the memory
15 device than was previously occupied by the first specified image data structure and that has a
16 lower associated image quality level than the image quality level associated with the first
17 specified image data structure; and
18 image reconstruction circuitry for successively applying a data decompression
19 method and an inverse transform to any specified one of the image data structures stored in the
20 memory device so as to generate a reconstructed image suitable for display on an image
21 viewer;
22 wherein the amount of space occupied by the image data structures in the memory
23 device can be reduced so as to make room for the storage of additional image data structures in
24 the memory device.

1 8. The image processing apparatus of claim 7, further including a communications
2 interface for receiving the image data from another apparatus.

1 9. The image processing apparatus of claim 8, wherein

2 each of a subset of the image data structures contains image transform data organized
3 on a transform layer basis such that image transform data for at least one transform layer is
4 stored in distinct portions of the image data structure from image transform data for other
5 transform layers; and

6 the image size reduction circuitry and one or more state machines includes logic,
7 operative when the first specified data structure is a member of the subset of image data
8 structures, for extracting a portion of the first specified image data structure that excludes the
9 image transform data for at least one transform layer and for replacing the first specified image
10 data structure with an image data structure containing the extracted portion.

1 10. The image processing apparatus of claim 8, wherein

2 each of a subset of the image data structures contains image transform data organized
3 on a transform layer basis such that image transform data for at least one transform layer is
4 stored in distinct portions of the image data structure from image transform data for other
5 transform layers; and

6 the image size reduction circuitry and one or more state machines includes logic,
7 operative when the first specified data structure is a member of the subset of image data
8 structures, for extracting a portion of the first specified image data structure that excludes the
9 image transform data for at least one transform layer and for replacing the first specified image
10 data structure with an image data structure containing the extracted portion.

1 11. Image processing apparatus, comprising:

2 a communications interface for receiving an image data structure having an associated
3 image quality level selected from a predefined range of image quality levels that range from a
4 highest quality level to a lowest quality level and that include at least two distinct quality
5 levels;

6 a memory device for storing the image data structure and other image data structures
7 representing a set of images;

8 image management logic, including:

9 image size reduction circuitry for extracting a subset of the data in a first
10 specified one of the image data structures to form a reduced size image data structure that
11 occupies less space in the memory device than was previously occupied by the first specified

12 image data structure and that has a lower associated image quality level than the quality level
13 associated with the first specified image data structure; and
14 image reconstruction circuitry for successively applying a data decompression
15 method and an inverse transform to any specified one of the image data structures and the
16 reduced size image data structure so as to generate a reconstructed image suitable for display
17 on a display device;
18 wherein the amount of space occupied by the image data structure in the memory
19 device can be reduced so as to make room for the storage of additional image data structures in
20 the memory device.

1 12. The image processing apparatus of claim 11, wherein
2 each image data structure contains image transform data organized on a bit plane basis
3 such that image transform data for at least one bit plane is stored in distinct portions of the
4 image data structure from image transform data for other bit planes;
5 the image size reduction circuitry and one or more state machines including logic for
6 extracting a portion of the first specified image data structure that excludes the image
7 transform data for at least one bit plane and for replacing the first specified image data
8 structure with an image data structure containing the extracted portion.

1 13. The image processing apparatus of claim 8, wherein
2 each of a subset of the image data structures contains image transform data organized
3 on a transform layer basis such that image transform data for at least one transform layer is
4 stored in distinct portions of the image data structure from image transform data for other
5 transform layers; and
6 the image size reduction circuitry and one or more state machines includes logic,
7 operative when the first specified data structure is a member of the subset of image data
8 structures, for extracting a portion of the first specified image data structure that excludes the
9 image transform data for at least one transform layer and for replacing the first specified image
10 data structure with an image data structure containing the extracted portion.

11 14. Image processing apparatus, the image processing apparatus comprising:
12 a communications interface for receiving an image data structure having an associated
13 image quality level selected from a predefined range of image quality levels that range from a

14 highest quality level to a lowest quality level and that include at least two distinct quality
15 levels;

16 a memory device for storing the image data structure and other image data structures
17 representing a set of images;

18 a data processor coupled to the memory device;

19 image management procedures, executable by the data processor, including instructions
20 for storing and processing image data, the instructions including:

21 an image size reduction procedure for lowering the quality level of a first
22 specified one of the image data structures, including instructions for extracting a subset of the
23 data in the first specified image data structure and forming a lower quality version of the first
24 specified image data structure that occupies less space in the memory device than was
25 previously occupied by the first specified image data structure; and

26 at least one image reconstruction procedure for successively applying a data
27 decompression method and an inverse transform to any specified one of the image data
28 structures stored in the memory device so as to generate a reconstructed image suitable for
29 display on an image viewer;

30 wherein the amount of space occupied by images stored in the form of image data
31 structures in the memory device can be reduced so as to make room for the storage of
32 additional image data structures in the memory device.

1 15. The image processing apparatus of claim 14, wherein
2 each of the image data structures contains image transform data organized on a bit
3 plane basis such that image transform data for at least one bit plane is stored in distinct
4 portions of the image data structure from image transform data for other bit planes; and
5 the image size reduction instructions include instructions for extracting a portion of an
6 image data structure that excludes the image transform data for at least one bit plane and for
7 replacing the image data structure with an image data structure containing the extracted
8 portion.

1 16. The image processing apparatus of claim 14, wherein
2 each of a subset of the image data structures contains image transform data organized
3 on a transform layer basis such that image transform data for at least one transform layer is

4 stored in distinct portions of the image data structure from image transform data for other
5 transform layers; and

6 the image size reduction instructions include instructions, operative when the first
7 specified data structure is a member of the subset of image data structures, for extracting a
8 portion of the first specified image data structure that excludes the image transform data for at
9 least one transform layer and for replacing the first specified image data structure with an
10 image data structure containing the extracted portion.

11 17. A computer program product, for use in conjunction with a computer system having a
12 memory in which image data structures can be stored, the computer program product
13 comprising a computer readable storage medium and a computer program mechanism
14 embedded therein, the computer program mechanism comprising:

15 an image handling procedure, including instructions for storing in the memory of the
16 computer system a plurality of image data structures,

17 an image size reduction procedure for accessing image data structures in the memory of
18 the computer system, each of the image data structures containing image transform data,
19 lowering the quality level of a first specified one of the image data structures, including
20 instructions for extracting a subset of the data in a first specified image data structure and
21 forming a lower quality version of the first specified image data structure that occupies less
22 space in the memory device than was previously occupied by the first specified image data
23 structure; and

24 at least one image reconstruction procedure for successively applying a data
25 decompression procedure and an inverse transform to any specified one of the image data
26 structures stored in the memory device so as to generate a reconstructed image suitable for
27 display on an image viewer;

28 wherein the amount of space occupied by images stored in the form of image data
29 structures in the memory device can be reduced so as to make room for the storage of
30 additional image data structures in the memory device.

1 18. The computer program product of claim 17, wherein

2 each of the image data structures contains image transform data organized on a bit
3 plane basis such that image transform data for at least one bit plane is stored in distinct
4 portions of the image data structure from image transform data for other bit planes; and

5 the image size reduction procedure includes instructions for extracting a portion of the
6 first specified image data structure that excludes the image transform data for at least one bit
7 plane and for replacing the first specified image data structure with an image data structure
8 containing the extracted portion.

1 19. The computer program product of claim 17, wherein
2 each of a subset of the image data structures contains image transform data organized
3 on a transform layer basis such that image transform data for at least one transform layer is
4 stored in distinct portions of the image data structure from image transform data for other
5 transform layers; and

6 the image size reduction procedure includes instructions, operative when the first
7 specified data structure is a member of the subset of image data structures, for extracting a
8 portion of the first specified image data structure that excludes the image transform data for at
9 least one transform layer and for replacing the first specified image data structure with an
10 image data structure containing the extracted portion.

1 20. The computer program product of claim 17, wherein the image handling procedure
2 includes one or more image processing procedures for applying a predefined transform to raw
3 image data to generate transform image data and for applying a data compression procedure to
4 the transform image data so as to generate an image data structure having an associated image
5 quality level selected from the predefined range of image quality levels.

1 21. A method of processing images, comprising:
2 storing in a memory device a plurality of image data structures that each represent a
3 respective image, each image data structure having an associated image quality level
4 corresponding to a quality level at which the corresponding image has been encoded in the
5 image data structure; the image quality level of each image data structure being a member of
6 predefined range of image quality levels that range from a highest quality level to a lowest
7 quality level and that include at least two distinct quality levels;

8 reducing the size of a specified one of the image data structures stored in the
9 nonvolatile memory device, including extracting a subset of the data in the specified image
10 data structure and forming a lower quality version of the specified image data structure that

11 occupies less space in the nonvolatile memory device than was previously occupied by the
12 specified image data structure; and

13 successively applying a data decompression method and an inverse transform to a
14 specified one of the image data structures stored in the nonvolatile memory device so as to
15 generate a reconstructed image suitable for display on an image viewer;

16 wherein the amount of space occupied by images stored in the form of image data
17 structures in the nonvolatile memory device can be reduced so as to make room for the storage
18 of additional image data structures in the nonvolatile memory device.

1 22. The method of claim 21, wherein the method is performed by a digital camera and the
2 method includes applying a predefined transform to image data received from an image capture
3 mechanism in the digital camera to generate transform image data, applying a data
4 compression method to the transform image data so as to generate an image data structure
5 having an associated image quality level selected from the predefined range of image quality
6 levels, and storing the image data structure in the memory device.

1 23. The method of claim 21, wherein the method includes applying a predefined transform
2 to raw image data to generate transform image data, applying a data compression method to the
3 transform image data so as to generate an image data structure having an associated image
4 quality level selected from the predefined range of image quality levels, and storing the image
5 data structure in the memory device.

1 24. The method of claim 21, wherein
2 each image data structure contains image transform data organized on a bit plane basis
3 such that image transform data for at least one bit plane is stored in distinct portions of the
4 image data structure from image transform data for other bit planes;
5 the size reduction step includes extracting a portion of an image data structure that
6 excludes the image transform data for at least one bit plane and for replacing the image data
7 structure in the nonvolatile memory device with an image data structure containing the
8 extracted portion.

1 25. The method of claim 21, wherein

2 each of a subset of the image data structures contains image transform data organized
3 on a transform layer basis such that image transform data for at least one transform layer is
4 stored in distinct portions of the image data structure from image transform data for other
5 transform layers; and

6 the size reduction step includes extracting a portion of the first specified image data
7 structure that excludes the image transform data for at least one transform layer and replacing
8 the first specified image data structure with an image data structure containing the extracted
9 portion.

1 26. Video image processing apparatus, comprising:

2 a memory device for storing a set of image data structures representing a sequence of
3 video frames, the set of image data structures having an associated image quality level selected
4 from a predefined range of image quality levels that range from a highest quality level to a
5 lowest quality level and that include at least two distinct quality levels; and

6 image management logic including:

7 image size reduction circuitry for extracting a subset of the data in the set of
8 image data structures and forming a lower quality version of the set of image data structures
9 that occupies less space in the memory device than was previously occupied by the set of
10 image data structures; and

11 image reconstruction circuitry for successively applying a data decompression
12 method and an inverse transform to at least a subset of the image data structures so as to
13 generate a reconstructed sequence of video frames suitable for display on a display device;

14 whereby the amount of space occupied by the set of image data structures in the
15 memory device can be reduced so as to make room for the storage of additional image data
16 structures in the memory device.

1 27. The video image processing apparatus of claim 26, wherein

2 each image data structure contains image transform data organized on a bit plane basis
3 such that image transform data for at least one bit plane is stored in distinct portions of the
4 image data structure from image transform data for other bit planes;

5 the image size reduction circuitry and one or more state machines including logic for
6 extracting a portion of an image data structure that excludes the image transform data for at

7 least one bit plane and for replacing the image data structure with an image data structure
8 containing the extracted portion.

1 28. The video image processing apparatus of claim 26, wherein
2 the image data structures contains image transform data organized on a transform layer
3 basis such that image transform data for at least one transform layer is stored in distinct
4 portions of the image data structure from image transform data for other transform layers; and
5 the image size reduction circuitry and one or more state machines includes logic,
6 operative when the first specified data structure is a member of the subset of image data
7 structures, for extracting a portion of the first specified image data structure that excludes the
8 image transform data for at least one transform layer and for replacing the first specified image
9 data structure with an image data structure containing the extracted portion.

1 29. The video image processing apparatus of claim 26, wherein the video frames are
2 divided into sub-sequences of N frames, where N is an integer greater than three, and the
3 predefined transform applied to the sequence of video images is a wavelet-like transform that
4 is applied to at least one video frame in each said sub-sequence of N frames.

1 30. The video image processing apparatus of claim 29, wherein the wavelet-like transform
2 is applied to at least one difference frame for each said sub-sequence of N frames, the
3 difference frame representing differences between one frame and a next frame in said sub-
4 sequence of N frames.

1 31. The video image processing apparatus of claim 26, wherein the video frames are
2 divided into sub-sequences of N frames, where N is an integer greater than three, and the
3 predefined transform applied to the sequence of video images is a wavelet-like transform that
4 is applied to separately and in time order to data at each x,y position in the video frames.

1 32. Video image processing apparatus, comprising:
2 image management logic, including:
3 image processing circuitry for applying a predefined transform to a sequence of
4 video frames to generate transform image data and for applying a data compression method to
5 the transform image data so as to generate a set of image data structures having an associated

6 image quality level selected from a predefined range of image quality levels that range from a
7 highest quality level to a lowest quality level and that include at least two distinct quality
8 levels;

9 a memory device for storing the set of image data structures;

10 the image management logic further including:

11 image size reduction circuitry for extracting a subset of the data in the set of
12 image data structures and forming a lower quality version of the set of image data structures
13 that occupies less space in the memory device than was previously occupied by the set of
14 image data structures; and

15 image reconstruction circuitry for successively applying a data decompression
16 method and an inverse transform to at least a subset of the image data structures so as to
17 generate a reconstructed sequence of video frames suitable for display on a display device;

18 whereby the amount of space occupied by the set of image data structures in the
19 memory device can be reduced so as to make room for the storage of additional image data
20 structures in the memory device.

1 33. The video image processing apparatus of claim 32, wherein
2 each image data structure contains image transform data organized on a bit plane basis
3 such that image transform data for at least one bit plane is stored in distinct portions of the
4 image data structure from image transform data for other bit planes;

5 the image size reduction circuitry and one or more state machines including logic for
6 extracting a portion of an image data structure that excludes the image transform data for at
7 least one bit plane and for replacing the image data structure with an image data structure
8 containing the extracted portion.

1 34. The video image processing apparatus of claim 32, wherein
2 the image data structures contains image transform data organized on a transform layer
3 basis such that image transform data for at least one transform layer is stored in distinct
4 portions of the image data structure from image transform data for other transform layers; and

5 the image size reduction circuitry and one or more state machines includes logic,
6 operative when the first specified data structure is a member of the subset of image data
7 structures, for extracting a portion of the first specified image data structure that excludes the

8 image transform data for at least one transform layer and for replacing the first specified image
9 data structure with an image data structure containing the extracted portion.

1 35. The video image processing apparatus of claim 32, wherein the video frames are
2 divided into sub-sequences of N frames, where N is an integer greater than three, and the
3 predefined transform applied to the sequence of video images is a wavelet-like transform that
4 is applied to at least one video frame in each said sub-sequence of N frames.

1 36. The video image processing apparatus of claim 35, wherein the wavelet-like transform
2 is applied to at least one difference frame for each said sub-sequence of N frames, the
3 difference frame representing differences between one frame and a next frame in said sub-
4 sequence of N frames.

1 37. The video image processing apparatus of claim 32, wherein the video frames are
2 divided into sub-sequences of N frames, where N is an integer greater than three, and the
3 predefined transform applied to the sequence of video images is a wavelet-like transform that
4 is applied to separately and in time order to data at each x,y position in the video frames.

1 38. Video image processing apparatus, comprising:
2 a memory device for storing a set of image data structures representing a sequence of
3 video frames, the set of image data structures having an associated image quality level selected
4 from a predefined range of image quality levels that range from a highest quality level to a
5 lowest quality level and that include at least two distinct quality levels;
6 a data processor coupled to the memory device;
7 image management procedures, executable by the data processor, including
8 an image size reduction procedure for extracting a subset of the data in the set
9 of image data structures and forming a lower quality version of the set of image data structures
10 that occupies less space in the memory device than was previously occupied by the set of
11 image data structures; and
12 at least one image reconstruction procedure for successively applying a data
13 decompression method and an inverse transform to at least a subset of the image data
14 structures so as to generate a reconstructed sequence of video frames suitable for display on a
15 display device;

16 whereby the amount of space occupied by the set of image data structures in the
17 memory device can be reduced so as to make room for the storage of additional image data
18 structures in the memory device.

1 39. The video image processing apparatus of claim 38, wherein
2 each image data structure contains image transform data organized on a bit plane basis
3 such that image transform data for at least one bit plane is stored in distinct portions of the
4 image data structure from image transform data for other bit planes;
5 the at least one image size reduction procedure includes instructions for extracting a
6 portion of an image data structure that excludes the image transform data for at least one bit
7 plane and for replacing the image data structure with an image data structure containing the
8 extracted portion.

1 40. The video image processing apparatus of claim 38, wherein
2 the image data structures contains image transform data organized on a transform layer
3 basis such that image transform data for at least one transform layer is stored in distinct
4 portions of the image data structure from image transform data for other transform layers; and
5 the at least one image size reduction procedure and one or more state machines include
6 logic, operative when the first specified data structure is a member of the subset of image data
7 structures, for extracting a portion of the first specified image data structure that excludes the
8 image transform data for at least one transform layer and for replacing the first specified image
9 data structure with an image data structure containing the extracted portion.

1 41. The video image processing apparatus of claim 38, wherein the video frames are
2 divided into sub-sequences of N frames, where N is an integer greater than three, and the
3 predefined transform applied to the sequence of video images is a wavelet-like transform that
4 is applied to at least one video frame in each said sub-sequence of N frames.

1 42. The video image processing apparatus of claim 41, wherein the wavelet-like transform
2 is applied to at least one difference frame for each said sub-sequence of N frames, the
3 difference frame representing differences between one frame and a next frame in said sub-
4 sequence of N frames.

- 1 43. The video image processing apparatus of claim 38, wherein the video frames are
2 divided into sub-sequences of N frames, where N is an integer greater than three, and the
3 predefined transform applied to the sequence of video images is a wavelet-like transform that
4 is applied to separately and in time order to data at each x,y position in the video frames.
- 1 44. Video image processing apparatus, comprising:
2 a memory device for storing image data structures;
3 a data processor coupled to the memory device;
4 image management procedures, executable by the data processor, including:
5 at least one image processing procedure for applying a predefined transform to a
6 sequence of video frames to generate transform image data and for applying a data
7 compression method to the transform image data so as to generate a set of image data
8 structures having an associated image quality level selected from a predefined range of image
9 quality levels that range from a highest quality level to a lowest quality level and that include
10 at least two distinct quality levels, and for storing the set of image data structures in the
11 memory device;
12 an image size reduction procedure for extracting a subset of the data in the set
13 of image data structures and forming a lower quality version of the set of image data structures
14 that occupies less space in the memory device than was previously occupied by the set of
15 image data structures; and
16 at least one image reconstruction procedure for successively applying a data
17 decompression method and an inverse transform to at least a subset of the image data
18 structures so as to generate a reconstructed sequence of video frames suitable for display on a
19 display device;
20 whereby the amount of space occupied by the set of image data structures in the
21 memory device can be reduced so as to make room for the storage of additional image data
22 structures in the memory device.

1 45. The video image processing apparatus of claim 44, wherein
2 each image data structure contains image transform data organized on a bit plane basis
3 such that image transform data for at least one bit plane is stored in distinct portions of the
4 image data structure from image transform data for other bit planes;

5 the at least one image size reduction procedure includes instructions for extracting a
6 portion of an image data structure that excludes the image transform data for at least one bit
7 plane and for replacing the image data structure with an image data structure containing the
8 extracted portion.

1 46. The video image processing apparatus of claim 44, wherein
2 the image data structures contains image transform data organized on a transform layer
3 basis such that image transform data for at least one transform layer is stored in distinct
4 portions of the image data structure from image transform data for other transform layers; and

5 the at least one image size reduction procedure and one or more state machines include
6 logic, operative when the first specified data structure is a member of the subset of image data
7 structures, for extracting a portion of the first specified image data structure that excludes the
8 image transform data for at least one transform layer and for replacing the first specified image
9 data structure with an image data structure containing the extracted portion.

10 47. The video image processing apparatus of claim 44, wherein the video frames are
11 divided into sub-sequences of N frames, where N is an integer greater than three, and the
12 predefined transform applied to the sequence of video images is a wavelet-like transform that
13 is applied to at least one video frame in each said sub-sequence of N frames.

1 48. The video image processing apparatus of claim 47, wherein the wavelet-like transform
2 is applied to at least one difference frame for each said sub-sequence of N frames, the
3 difference frame representing differences between one frame and a next frame in said sub-
4 sequence of N frames.

1 49. The video image processing apparatus of claim 44, wherein the video frames are
2 divided into sub-sequences of N frames, where N is an integer greater than three, and the
3 predefined transform applied to the sequence of video images is a wavelet-like transform that
4 is applied to separately and in time order to data at each x,y position in the video frames.

5 50. A computer program product, for use in conjunction with a computer system having a
6 memory in which image data structures can be stored, the computer program product
7 comprising a computer readable storage medium and a computer program mechanism
8 embedded therein, the computer program mechanism comprising:

9 an image handling procedure, including instructions for storing in the memory of the
10 computer system a set of image data structures representing a sequence of video frames, the set
11 of image data structures having an associated image quality level selected from a predefined
12 range of image quality levels that range from a highest quality level to a lowest quality level
13 and that include at least two distinct quality levels;

14 a data processor coupled to the memory device;

15 an image size reduction procedure for extracting a subset of the data in the set of image
16 data structures and forming a lower quality version of the set of image data structures that
17 occupies less space in the memory device than was previously occupied by the set of image
18 data structures; and

19 at least one image reconstruction procedure for successively applying a data
20 decompression method and an inverse transform to at least a subset of the image data
21 structures so as to generate a reconstructed sequence of video frames suitable for display on a
22 display device;

23 whereby the amount of space occupied by the set of image data structures in the
24 memory device can be reduced so as to make room for the storage of additional image data
25 structures in the memory device.

1 51. The computer program product of claim 50, wherein
2 each image data structure contains image transform data organized on a bit plane basis
3 such that image transform data for at least one bit plane is stored in distinct portions of the
4 image data structure from image transform data for other bit planes;

5 the at least one image size reduction procedure includes instructions for extracting a
6 portion of an image data structure that excludes the image transform data for at least one bit

7 plane and for replacing the image data structure with an image data structure containing the
8 extracted portion.

1 52. The computer program product of claim 50, wherein
2 the image data structures contains image transform data organized on a transform layer
3 basis such that image transform data for at least one transform layer is stored in distinct
4 portions of the image data structure from image transform data for other transform layers; and
5 the at least one image size reduction procedure and one or more state machines include
6 logic, operative when the first specified data structure is a member of the subset of image data
7 structures, for extracting a portion of the first specified image data structure that excludes the
8 image transform data for at least one transform layer and for replacing the first specified image
9 data structure with an image data structure containing the extracted portion.

1 53. The computer program product of claim 50, wherein the video frames are divided into
2 sub-sequences of N frames, where N is an integer greater than three, and the predefined
3 transform applied to the sequence of video images is a wavelet-like transform that is applied to
4 at least one video frame in each said sub-sequence of N frames.

1 54. The computer program product of claim 53, wherein the wavelet-like transform is
2 applied to at least one difference frame for each said sub-sequence of N frames, the difference
3 frame representing differences between one frame and a next frame in said sub-sequence of N
4 frames.

1 55. The computer program product of claim 50, wherein the video frames are divided into
2 sub-sequences of N frames, where N is an integer greater than three, and the predefined
3 transform applied to the sequence of video images is a wavelet-like transform that is applied to
4 separately and in time order to data at each x,y position in the video frames.

1 56. The computer program product of claim 50, including:
2 at least one image processing procedure for applying a predefined transform to a
3 sequence of video frames to generate transform image data and for applying a data
4 compression method to the transform image data so as to generate the set of image data
5 structures stored in the memory.

1 57. The computer program product of claim 56, wherein
2 each image data structure contains image transform data organized on a bit plane basis
3 such that image transform data for at least one bit plane is stored in distinct portions of the
4 image data structure from image transform data for other bit planes;
5 the at least one image size reduction procedure includes instructions for extracting a
6 portion of an image data structure that excludes the image transform data for at least one bit
7 plane and for replacing the image data structure with an image data structure containing the
8 extracted portion.

1 58. The computer program product of claim 56, wherein
2 the image data structures contains image transform data organized on a transform layer
3 basis such that image transform data for at least one transform layer is stored in distinct
4 portions of the image data structure from image transform data for other transform layers; and
5 the at least one image size reduction procedure and one or more state machines include
6 logic, operative when the first specified data structure is a member of the subset of image data
7 structures, for extracting a portion of the first specified image data structure that excludes the
8 image transform data for at least one transform layer and for replacing the first specified image
9 data structure with an image data structure containing the extracted portion.

1 59. The computer program product of claim 56, wherein the video frames are divided into
2 sub-sequences of N frames, where N is an integer greater than three, and the predefined
3 transform applied to the sequence of video images is a wavelet-like transform that is applied to
4 at least one video frame in each said sub-sequence of N frames.

1 60. The computer program product of claim 59, wherein the wavelet-like transform is
2 applied to at least one difference frame for each said sub-sequence of N frames, the difference
3 frame representing differences between one frame and a next frame in said sub-sequence of N
4 frames.

1 61. The computer program product of claim 56, wherein the video frames are divided into
2 sub-sequences of N frames, where N is an integer greater than three, and the predefined
3 transform applied to the sequence of video images is a wavelet-like transform that is applied to
4 separately and in time order to data at each x,y position in the video frames.

1 62. A method of processing video images, comprising:
2 storing in a memory device a set of image data structures representing a sequence of
3 video frames, the set of image data structures having an associated image quality level selected
4 from a predefined range of image quality levels that range from a highest quality level to a
5 lowest quality level and that include at least two distinct quality levels;

6 extracting a subset of the data in the set of image data structures and forming a lower
7 quality version of the set of image data structures that occupies less space in the memory
8 device than was previously occupied by the set of image data structures; and

9 successively applying a data decompression method and an inverse transform to the
10 specified set of image data structures so as to generate a reconstructed sequence of video
11 images suitable for display on a display device;

12 whereby the amount of space occupied by the set of image data structures in the
13 memory device can be reduced so as to make room for the storage of additional image data
14 structures in the memory device.

1 63. The method of claim 62, wherein
2 each of the image data structures contains image transform data organized on a bit
3 plane basis such that image transform data for at least one bit plane is stored in distinct
4 portions of the image data structure from image transform data for other bit planes;
5 the extracting step includes extracting a portion of an image data structure that excludes
6 the image transform data for at least one bit plane, and the forming step includes replacing the
7 image data structure with an image data structure containing the extracted portion.

1 64. The method of claim 62, wherein
2 the of the image data structures contains image transform data organized on a transform
3 layer basis such that image transform data for at least one transform layer is stored in distinct
4 portions of the image data structure from image transform data for other transform layers; and

5 the extracting step includes extracting a portion of the first specified image data
6 structure that excludes the image transform data for at least one transform layer, and the
7 forming step includes replacing the first specified image data structure with an image data
8 structure containing the extracted portion.

1 65. The method of claim 62, after performing the extracting and forming steps, applying
2 the predefined transform to a sequence of additional video images to generate transform image
3 data and applying the data compression method to the transform image data so as to generate
4 an additional set of image data structures having an associated image quality, and storing the
5 additional set of image data structures in the memory device.

1 66. The method of claim 62, wherein the video frames are divided into sub-sequences of N
2 frames, where N is an integer greater than three, and the predefined transform applied to the
3 sequence of video images is a wavelet-like transform that is applied to at least one video frame
4 in each said sub-sequence of N frames.

1 67. The method of claim 66, wherein the wavelet-like transform is applied to at least one
2 difference frame for each said sub-sequence of N frames, the difference frame representing
3 differences between one frame and a next frame in said sub-sequence of N frames.

1 68. The method of claim 62, wherein the video frames are divided into sub-sequences of N
2 frames, where N is an integer greater than three, and the predefined transform applied to the
3 sequence of video images is a wavelet-like transform that is applied to separately and in time
4 order to data at each x,y position in the video frames.

1 69. The method of claim 62, including applying a predefined transform to a sequence of
2 video images to generate transform image data and applying a data compression method to the
3 transform image data so as to generate the set of image data structures stored in the memory
4 device.

1 70. The method of claim 69, wherein

2 each of the image data structures contains image transform data organized on a bit
3 plane basis such that image transform data for at least one bit plane is stored in distinct
4 portions of the image data structure from image transform data for other bit planes;

5 the extracting step includes extracting a portion of an image data structure that excludes
6 the image transform data for at least one bit plane, and the forming step includes replacing the
7 image data structure with an image data structure containing the extracted portion.

1 71. The method of claim 69, wherein

2 the of the image data structures contains image transform data organized on a transform
3 layer basis such that image transform data for at least one transform layer is stored in distinct
4 portions of the image data structure from image transform data for other transform layers; and

5 the extracting step includes extracting a portion of the first specified image data
6 structure that excludes the image transform data for at least one transform layer, and the
7 forming step includes replacing the first specified image data structure with an image data
8 structure containing the extracted portion.

1 72. The method of claim 69, after performing the extracting and forming steps, applying
2 the predefined transform to a sequence of additional video images to generate transform image
3 data and applying the data compression method to the transform image data so as to generate
4 an additional set of image data structures having an associated image quality, and storing the
5 additional set of image data structures in the memory device.

1 73. The method of claim 69, wherein the video frames are divided into sub-sequences of N
2 frames, where N is an integer greater than three, and the predefined transform applied to the
3 sequence of video images is a wavelet-like transform that is applied to at least one video frame
4 in each said sub-sequence of N frames.

1 74. The method of claim 73, wherein the wavelet-like transform is applied to at least one
2 difference frame for each said sub-sequence of N frames, the difference frame representing
3 differences between one frame and a next frame in said sub-sequence of N frames.

1 75. The method of claim 69, wherein the video frames are divided into sub-sequences of N
2 frames, where N is an integer greater than three, and the predefined transform applied to the
3 sequence of video images is a wavelet-like transform that is applied to separately and in time
4 order to data at each x,y position in the video frames.

1/7

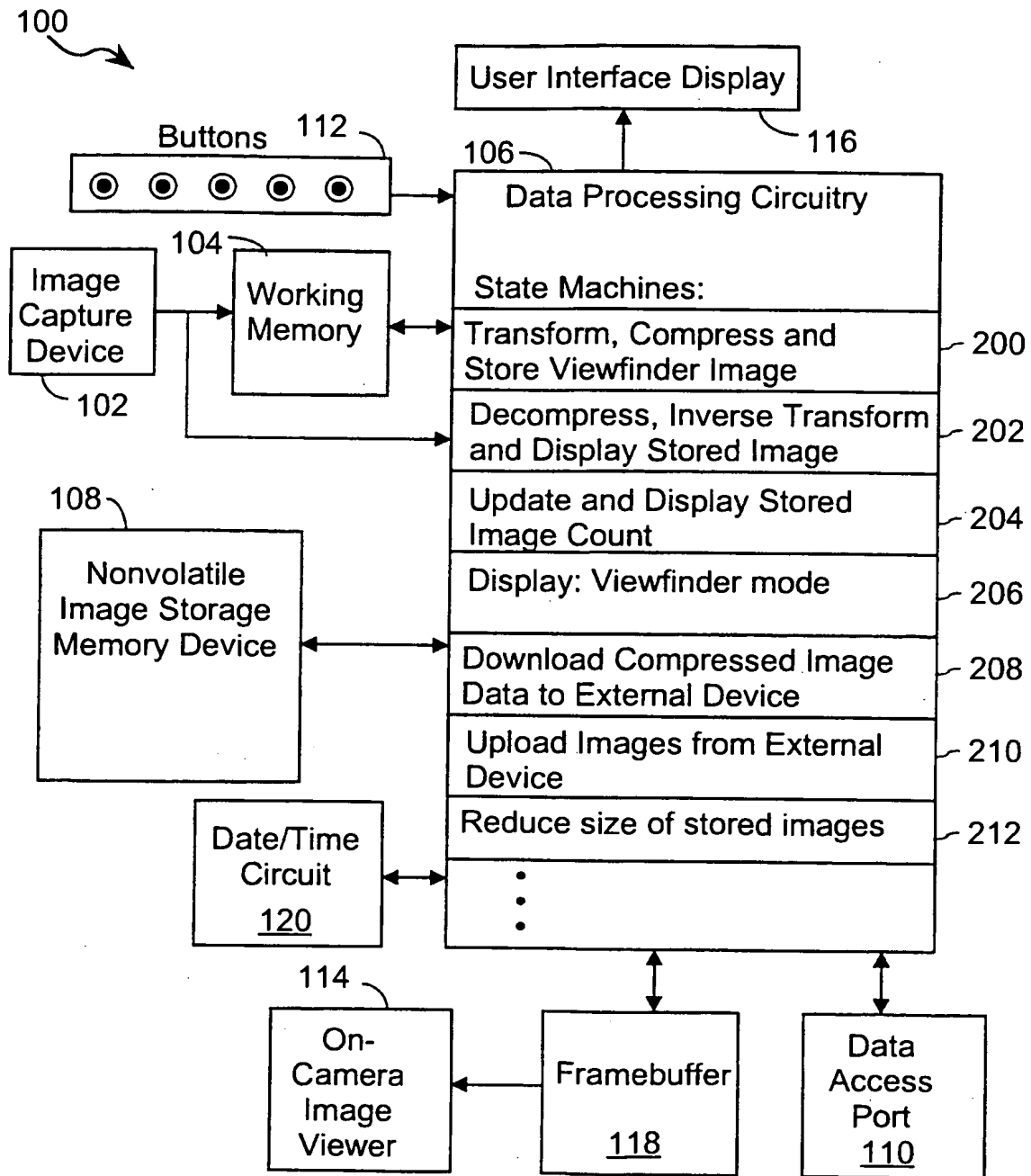


FIG. 1

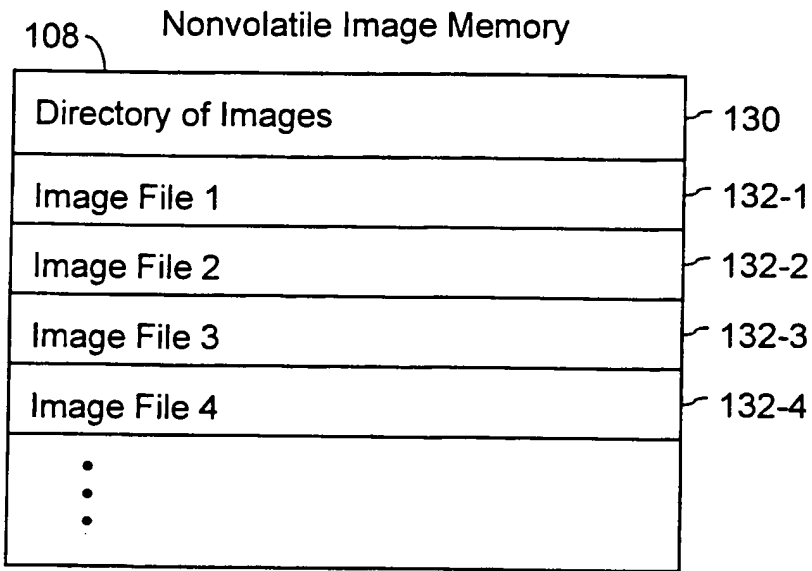


FIG. 2

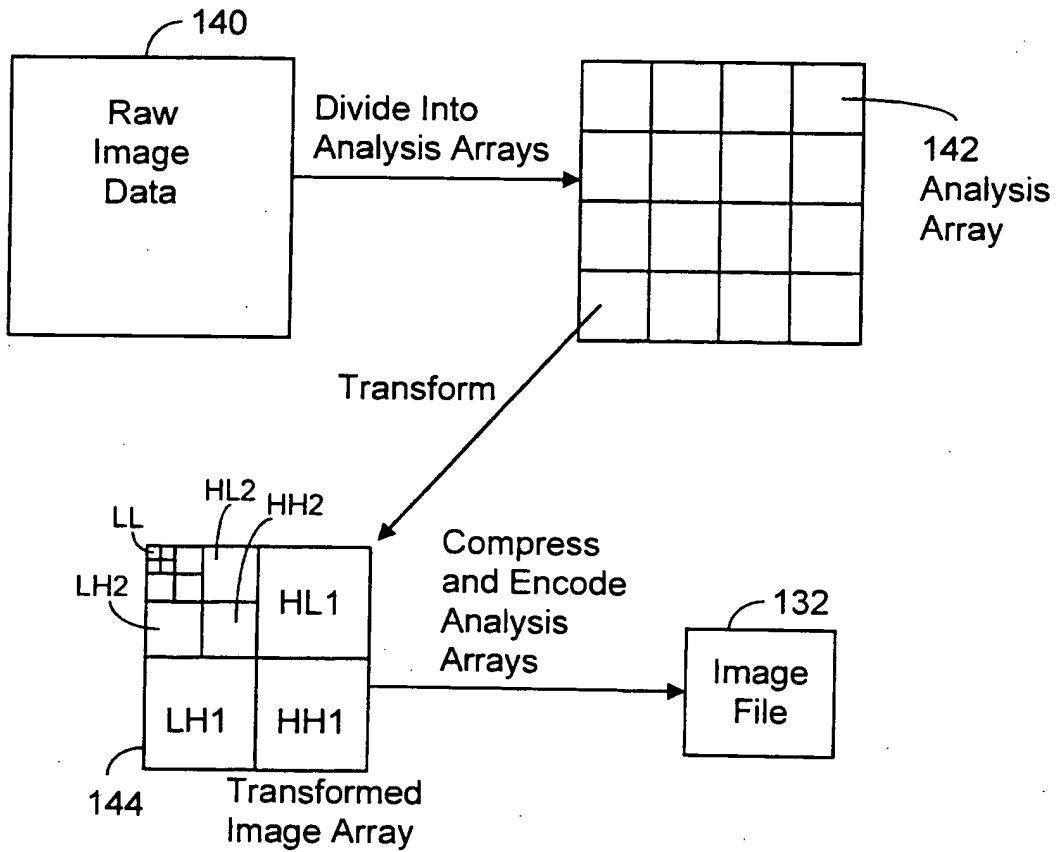


FIG. 3

Image File (Compressed Encoded Image Data Structure)

132

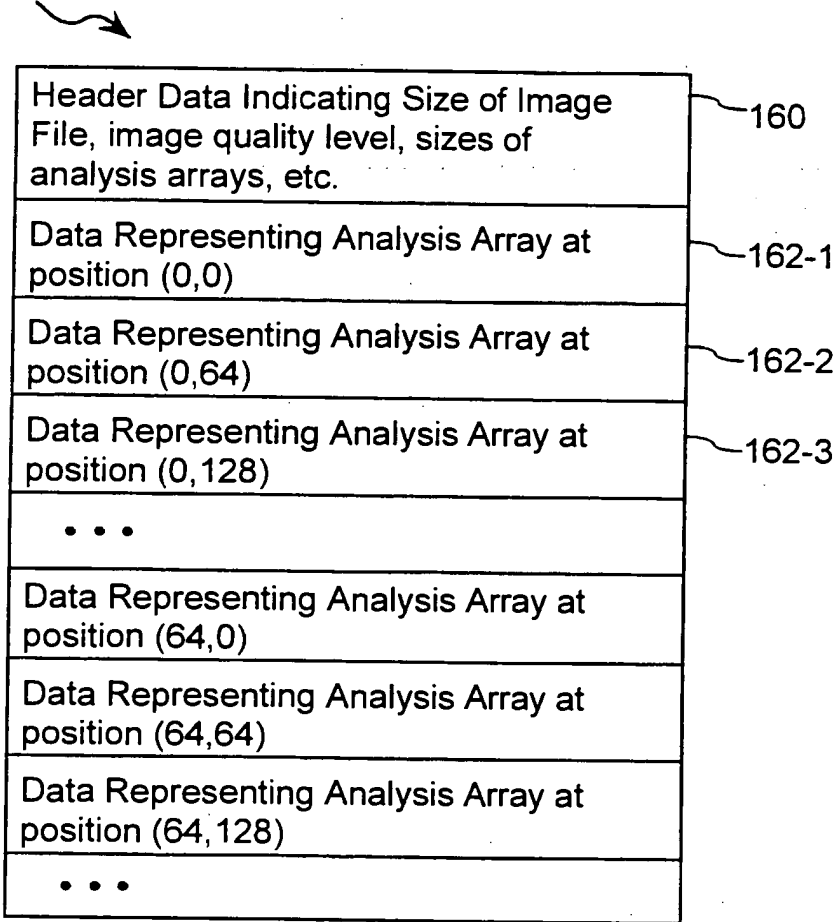


FIG. 4A

Data Representing One Analysis Array

162

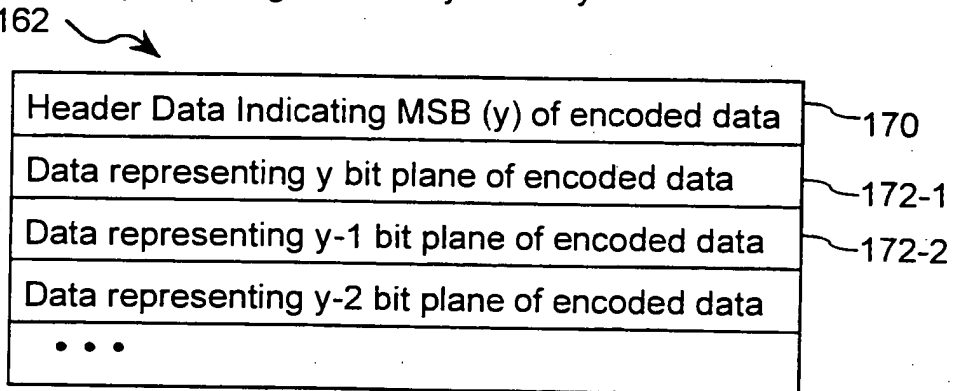


FIG. 4B

4/7

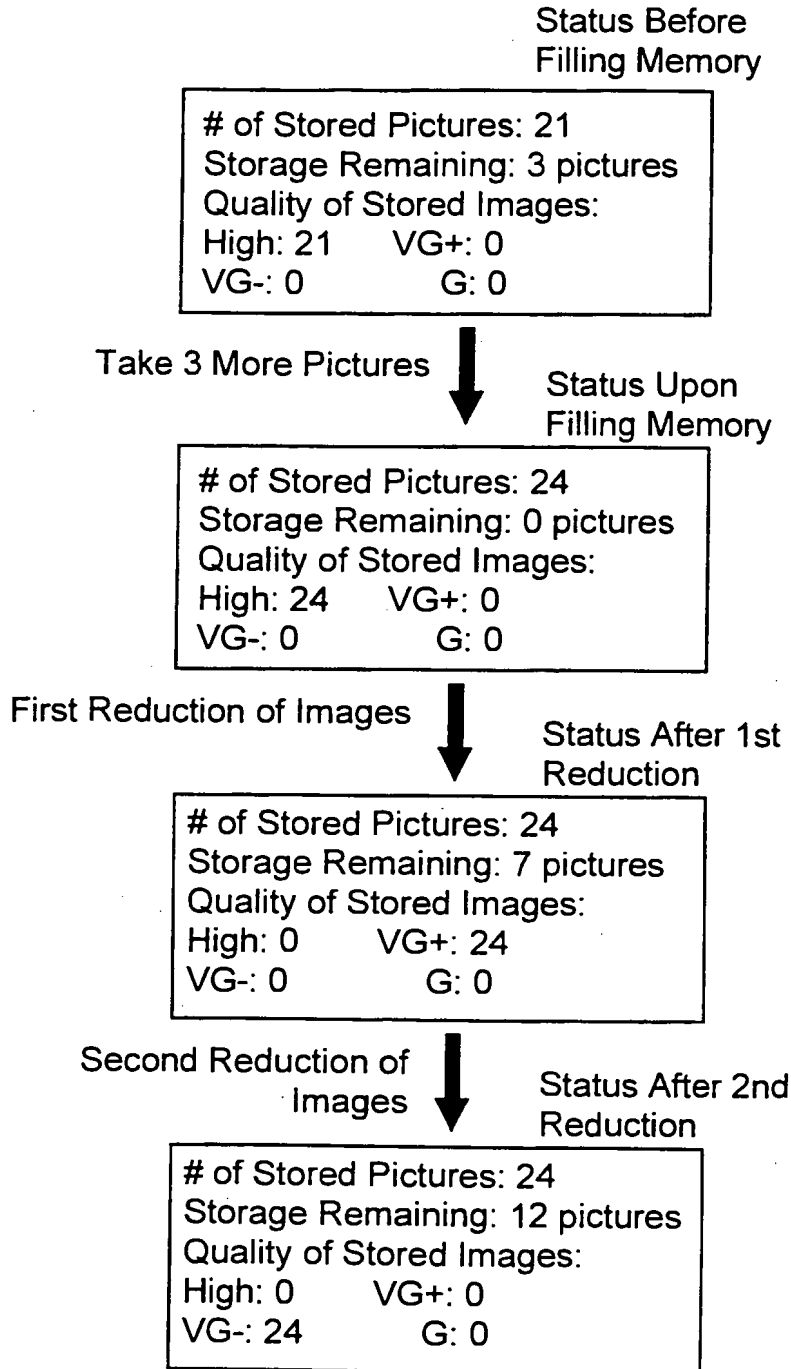


FIG. 5

5/7

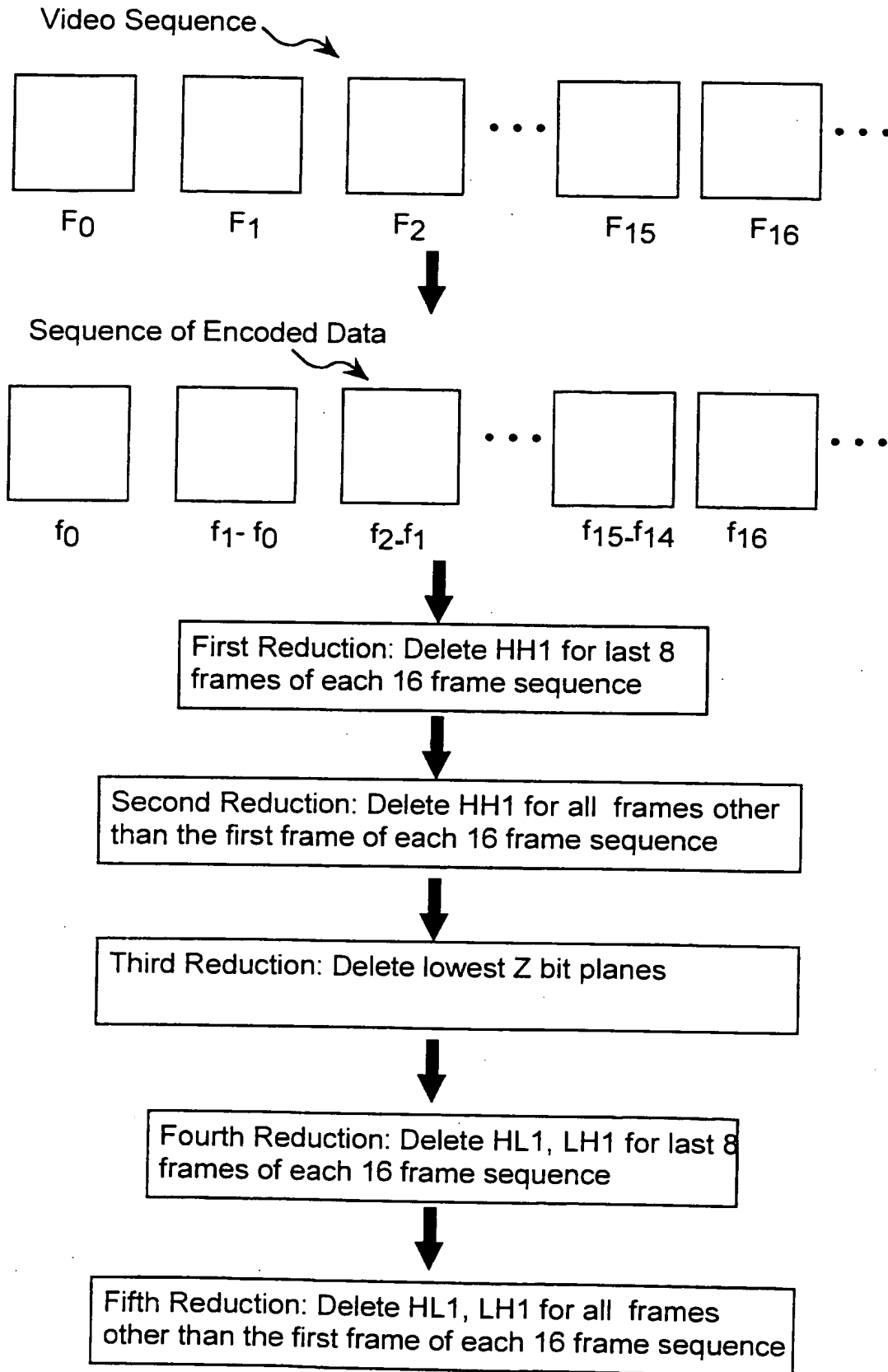


FIG. 6

6/7

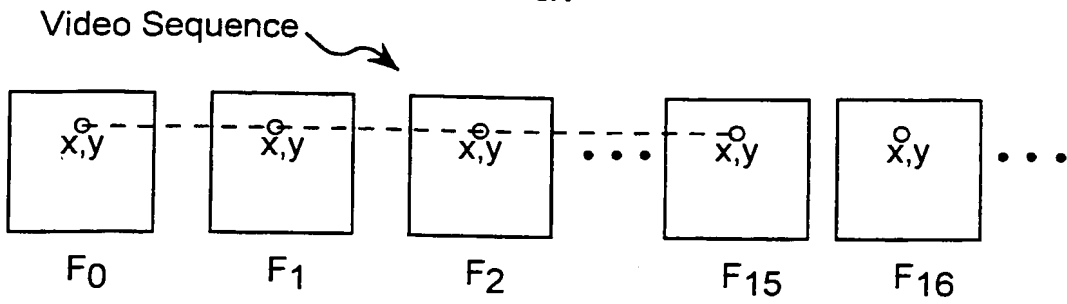


FIG. 7

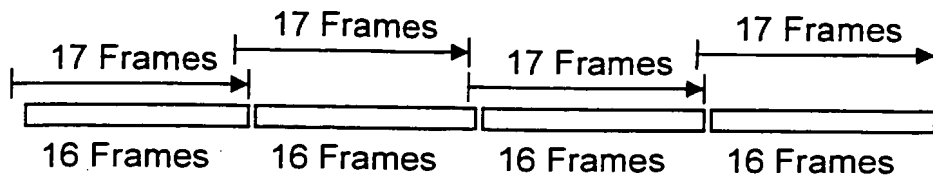


FIG. 8

Stored Coefficients for Time Dimension Wavelet Transform

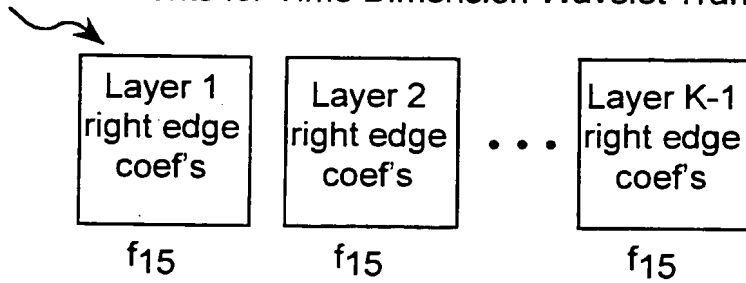


FIG. 9

717

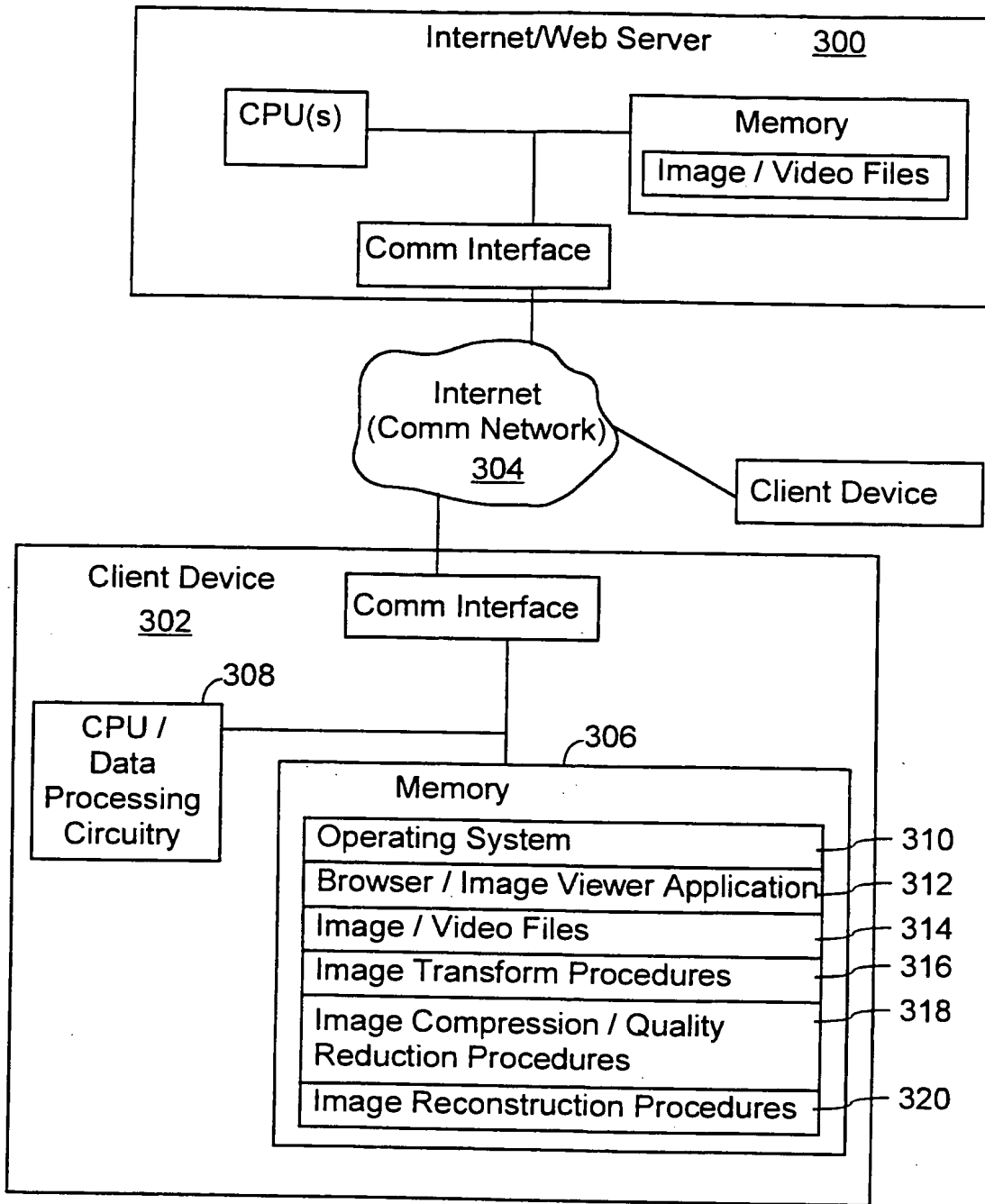


FIG. 10

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US00/30825

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06K 9/36, 9/46
 US CL : 382/232

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : Please See Continuation Sheet

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
 WEST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,867,602 A (ZANDI ET AL.) 02 February 1999 (02.02.99).	1-75
A	US 5,881,176 A (KEITH ET AL.) 09 March 1999 (09.03.1999).	1-75
A	US 5,966,465 A (KEITH ET AL.) 12 October, 1999 (12.10.1999).	1-75

Further documents are listed in the continuation of Box C. See patent family annex.

<p>* Special categories of cited documents:</p> <p>"A" document defining the general state of the art which is not considered to be of particular relevance</p> <p>"E" earlier application or patent published on or after the international filing date</p> <p>"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)</p> <p>"O" document referring to an oral disclosure, use, exhibition or other means</p> <p>"P" document published prior to the international filing date but later than the priority date claimed</p>	<p>"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention</p> <p>"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone</p> <p>"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art</p> <p>"&" document member of the same patent family</p>
---	---

Date of the actual completion of the international search 04 January 2001 (04.01.2001)	Date of mailing of the international search report 30 MAR 2001
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703)305-3230	Authorized officer Jose L. Couso Telephone No. (703)305-8576

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US00/30825

Continuation of B. FIELDS SEARCHED Item 1: 382/232, 233, 234, 240, 244, 246, 247, 248, 260, 263, 264, 276;
341/51, 63, 65, 67, 107; 364/724.011, 724.04, 724.05, 724.13, 724.14, 725.01, 725.02.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
7 December 2000 (07.12.2000)

PCT

(10) International Publication Number
WO 00/73922 A2

- (51) International Patent Classification⁷: G06F 17/00
- (21) International Application Number: PCT/US00/11078
- (22) International Filing Date: 25 April 2000 (25.04.2000)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
09/323,635 1 June 1999 (01.06.1999) US
- (71) Applicant: ENTERA, INC. [US/US]; 40971 Encyclopedia Circle, Fremont, CA 94538 (US).
- (72) Inventor: SCHARBER, John, M.; 1616 Placer Circle, Livermore, CA 94550 (US).
- (74) Agents: FAHMI, Tarek, N. et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).
- (81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.
- (84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— Without international search report and to be republished upon receipt of that report.

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.



WO 00/73922 A2

(54) Title: CONTENT DELIVERY SYSTEM

(57) Abstract: Disclosed is a network content delivery system configured to: select a first content routing technique for processing a first set of network content; and select a second content routing technique for processing a second set of network content, wherein the first and second content routing techniques are selected based on one or more content routing variables. Also disclosed is a content delivery system comprising: a network node for storing network content; a first transmission medium communicatively coupled to the network node for transmitting a first set of network content to the network node; and a second transmission medium communicatively coupled to the network node for transmitting a second set of network content to the network node, wherein the first and second sets of network content are selected based on one or more routing variables.

CONTENT DELIVERY SYSTEM

1

BACKGROUND OF THE INVENTION**Field of the Invention**

This invention relates to the transmission and storage of digital information across a network. More particularly, the invention relates to an improved system and method for caching and/or delivering various types of digital content using a plurality of network protocols.

Description of the Related Art

The World Wide Web (hereinafter "the Web") is a network paradigm which links documents known as "Web pages" locally or remotely across multiple network nodes (i.e., Web servers). A single Web page may have links (a.k.a., "hyperlinks") which point to numerous other Web pages. When a user points and clicks on a link using a cursor control device such as a mouse, the user can jump from the initial page to another page, regardless of where the Web pages are actually located. For example, the initial Web page might be stored on a Web server in New York and the second page (accessed via the hyperlink in the first page) might be stored on a Web server in California.

The underlying principles of the Web were developed 1989 at the European Center for Nuclear Research (CERN) in Geneva. By 1994 there were approximately 500 Web servers on the Internet. Today there are more than a million, with new sites starting up at an extraordinary rate. In sum, the Web has become the center of Internet activity and is the primary reason for the explosive growth of the Internet over the past several years.

In addition to providing a simple point-and-click interface to vast amounts of information on the Internet, the Web is quickly turning into a content delivery system. Well known Internet browsers such as Netscape Navigator[™] and Microsoft Internet Explorer[™] frequently provide plug-in software which allow additional features to be incorporated into the browser program. These include, for example, support for audio and video streaming, telephony, and videoconferencing.

The unparalleled increase in Web usage combined with the incorporation of high bandwidth applications (i.e., audio and video) into browser programs has created serious

performance/bandwidth problems for most Internet Service Providers (hereinafter "ISPs"). Moreover, the network traffic resulting from non-Web-based Internet services such as Internet News (commonly known as "Usenet" News) has increased on the same scale as the increase in Web traffic, thereby further adding to the bandwidth problems experienced by most ISPs.

These issues will be described in more detail with respect to **Figure 1** which illustrates an ISP 100 with a link 160 to a larger network 150 (e.g., the Internet) through which a plurality of clients 130, 120 can access a plurality of Web servers 140-144 and/or News servers 146-148. Maintaining a link 160 to the Internet 150 with enough bandwidth to handle the continually increasing traffic requirements of its clients 120, 130 represents a significant cost for ISP. At the same time, ISP 110 must absorb this cost in order to provide an adequate user experience for its clients 120, 130.

One system which is currently implemented to reduce network traffic across link 160 is a proxy server 210 with a Web cache 220, illustrated in **Figure 2**. When client 120 initially clicks on a hyperlink and requests a Web page (shown as address "www.isp.com/page.html") stored on Web server 144, client 120 will use proxy server 210 as a "proxy agent." This means that proxy server 210 will make the request for the Web page on behalf of client 120 as shown. Once the page has been retrieved and forwarded to client 120, proxy server will store a copy of the Web page locally in Web cache 220. Thus, when client 120 or another client – e.g., client 130 – makes a subsequent request for the same Web page, proxy server 210 will immediately transfer the Web page from its Web cache 210 to client 130. As a result, the speed with which client 130 receives the requested page is substantially increased, and at the same time, no additional bandwidth is consumed across Internet link 160.

While the foregoing proxy server configuration alleviates some of the network traffic across Internet link 160, several problems remain. One problem is that prior Web cache configurations do not have sufficient intelligence to deal with certain types of Web pages (or other Web-based information). For example, numerous Web pages and associated content can only be viewed by a client who pays a subscriber fee. As such, only those clients which provide proper authentication should be permitted to download the information. Today, proxy servers such as proxy server 210 will simply not cache a Web document which requires authentication.

In addition, Web caches do not address the increasing bandwidth problem associated with non-Web based Internet information. In particular, little has been done to alleviate the increasing bandwidth problems created by Usenet news streams. In fact, ISPs today must set aside a substantial amount of bandwidth to provide a continual Usenet news feed to its clients. Moreover, no mechanism is currently available for caching other data transmissions such as the streaming of digital audio and video. The term "streaming" implies a one-way transmission from a server to a client which provides for uninterrupted sound or video. When receiving a streaming transmission, the client will buffer a few seconds of audio or video information before it starts sending the information to a pair of speakers and/or a monitor, thus compensating for momentary delays in packet delivery across the network.

Accordingly, what is needed is a content delivery system which will reduce the bandwidth requirements for ISP 110 while still providing clients 120, 130 with an adequate user experience. What is also needed is a system which will work seamlessly with different types of Web-based and non-Web-based information and which can be implemented on currently available hardware and software platforms. What is also needed is an intelligent content delivery system which is capable of caching all types of Web-based information, including information which requires the authentication of a client before it can be accessed. What is also needed is a content delivery system which is easily adaptable so that it can be easily reconfigured to handle the caching of new Internet information and protocols. Finally, what is needed is a data replication system which runs on a distributed database engine, thereby incorporating well known distributed database procedures for maintaining cache coherency.

SUMMARY OF THE INVENTION

Disclosed is a network content delivery system configured to: select a first content routing technique for processing a first set of network content; and select a second content routing technique for processing a second set of network content, wherein the first and second content routing techniques are selected based on one or more content routing variables.

Also disclosed is a content delivery system comprising: a network node for storing network content; a first transmission medium communicatively coupled to the network node for transmitting a first set of network content to the network node; and a second transmission medium communicatively coupled to the network node for transmitting a second set of

network content to the network node, wherein the first and second sets of network content are selected based on one or more routing variables.

BRIEF DESCRIPTION OF THE DRAWINGS

A better understanding of the present invention can be obtained from the following detailed description in conjunction with the following drawings, in which:

FIG. 1 illustrates generally a network over which an ISP and a plurality of servers communicate.

FIG. 2 illustrates an ISP implementing a proxy server Web cache.

FIG. 3 illustrates one embodiment of the underlying architecture of an Internet content delivery system node.

FIG. 4 illustrates a plurality of Internet content delivery system nodes communicating across a network.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

One embodiment of the present system is a computer comprising a processor and a memory with which software implementing the functionality of the internet content delivery system described herein is executed. Such a computer system stores and communicates (internally or with other computer systems over a network) code and data using machine readable media, such as magnetic disks, random access memory, read only memory, carrier waves, signals, etc. In addition, while one embodiment is described in which the parts of the present invention are implemented in software, alternative embodiments can implement one or more of these parts using any combination of software, firmware and/or hardware.

The underlying architecture of one embodiment of the present internet content delivery system (hereinafter "ICDS") is illustrated in **Figure 3**. A single ICDS node 300 is shown including a cache 330, an ICDS application programming interface (hereinafter "APT") 360 which includes a distributed database engine 361, and a plurality of software modules 310-326

which interface with the ICDS API 360. ICDS node 300 may communicate over a network 340 (e.g., the Internet) over communication link 370 and may also interface with a plurality of clients 350-351 and/or other ICDS nodes (e.g., through link 380).

As is known in the art, an API such as ICDS API 360 is comprised of a plurality of subroutines which can be invoked by application software (i.e., software written to operate in conjunction with the particular API). Thus, in **Figure 3** each of the plurality of software modules 310-326 may be uniquely tailored to meet the specific needs of a particular ISP. The modules interface with API 360 by making calls to the API's set of predefined subroutines. Another significant feature of ICDS API 360 is that it is platform-independent. Accordingly, it can be implemented on numerous hardware platforms including those that are Intel-based, Macintosh-based and Sun Microsystems-based.

In one embodiment, a portion of API 360's subroutines and a set of prefabricated modules can be marketed together as a Software Development Kit (hereinafter "SDK"). This will allow ISPs, corporations and/or end-users to customize the type of internet content delivery/caching which they require. In addition, because modules 310-326 may be dynamically linked, they may be loaded and unloaded without having to reboot the hardware platform on which cache 330 is executed.

I. Distributed Content Processing

As illustrated, ICDS node 300 includes a plurality of network protocol modules 310-319 which interface with API 360. These modules provide caching support on ICDS node 300 for numerous different Internet protocols including, but not limited to, Web protocols such as the Hypertext Transfer Protocol (hereinafter "HTTP") 310, Usenet news protocols such as the Network News Transport Protocol (hereinafter "NNRP") 312, directory access protocols such as the Lightweight Directory Access Protocol (hereinafter "LDAP") 314, data streaming protocols such as the Real Time Streaming Protocol (hereinafter "RTSP") 316, and protocols used to perform Wide Area Load Balancing (hereinafter "WALB") 318. Because the underlying architecture of the present ICDS system includes an open API, new protocol modules (e.g., module 319) can be seamlessly added to the system as needed.

One embodiment of the ICDS system includes a plurality of standardized service definitions through which individual service modules 320-326 may be configured to interface

with the ICDS API 360. These service modules provide the underlying functionality of ICDS node 300 and may include a data services module 320, an access services module 322, a transaction services module 323, a commercial services module 324, a directory services module 325, and a resource services module 326. The functionality of each of these modules will be described in more detail below.

In one embodiment of the ICDS system, the ICDS API includes a distributed relational database engine 361. As a result, a plurality of ICDS nodes 410-440 can be distributed across ISP 400's internal network and still maintain a coherent, up-to-date storage of Internet content. For example, if a particular data object is updated at two nodes simultaneously, the underlying distributed database system may be configured to resolve any conflicts between the two modifications using a predefined set of distributed database algorithms. Accordingly, the present system provides built in caching support for dynamically changing Internet content (e.g., Web pages which are modified on a regular basis). Such a result was not attainable with the same level of efficiency in prior art caching systems such as proxy server 210 of **Figure 2** (which are executed on, e.g., standard flat file systems such as UNIX or NFS file servers).

Data Services

Data services modules such as module 320 running on each ICDS node 410-440 provide support for data replication and distribution across ISP 400's internal network 480. This includes caching support for any data protocol included in the set of protocol modules 310-319 shown in **Figure 3** as well as for any future protocol which may be added as a module to the ICDS API 360. Because the ICDS API 360 provides a set of standardized service definitions for data services module 320, an ISP using a plurality of ICDS nodes 300 as illustrated in **Figure 4** can replicate data across its network without an extensive knowledge of distributed database technology. In other words, the ISP can configure its plurality of nodes by invoking the standardized service definitions associated with data services module 320 and leave the distributed database functionality to the distributed database engine 361.

Generally, three different types of data replication may be implemented by the present system: dynamic replication, database replication (or "actual" replication), and index replication. Using dynamic replication, if client 472 requests content from internal ICDS server 460 or from a server across network 490, the content will be delivered to client 472 and replicated in ICDS node 430. If client 473 (or any other client) subsequently requests the

same content, it will be transmitted directly from ICDS node 430 rather than from its original source (i.e., a second request to server 460 or a server across network 490 will not be required). Accordingly, bandwidth across ISP 400's internal network and across Internet link 405 is conserved.

The dynamic replication mechanism just described works well for replicating static content but not for replicating dynamically changing content. For example, if the replicated content is a magazine article then caching a copy locally works well because it is static information – i.e., there is no chance that the local copy will become stale (out of date). However, if the replicated content is a Web page which contains continually changing information such as a page containing stock market quotes, then dynamic replication may not be appropriate. No built in mechanism is available for proxy cache server 210 to store an up-to-date copy of the information locally.

The present ICDS system, however, may use database replication to maintain up-to-date content at each ICDS node 410-440. Because the present system includes a distributed database engine 361, when a particular piece of content is changed at one node (e.g., ICDS server 460) a store procedure may be defined to update all copies of the information across the network. This may be in the form of a relational database query. Thus, the present system may be configured to use dynamic replication for static content but to use database replication for time-sensitive, dynamically changing content.

The third type of database replication is known as index replication. Using index replication a master index of content is replicated at one or more ICDS nodes 410-440 across the network 480. Once again, this implementation is simplified by the fact that the underlying ICDS node engine is a distributed database engine. Certain types of information distributions are particularly suitable for using index replication. For example, news overview information (i.e., the list of news articles in a particular newsgroup) is particularly suited to index replication. Instead of replicating each individual article, only the news overview information needs to be replicated at various nodes 410-440 across the network 480. When a client 473 wants to view a particular article, only then will the article be retrieved and cached locally (e.g., on ICDS node 430).

ICDS node 430 is capable of caching and delivering various types of Internet data using any of the foregoing replication techniques. While prior art proxy servers such as proxy server 210 may only be used for caching Web pages, ICDS node 430 is capable of caching various other types of internet content (e.g., news content) as a result of the protocol modules 310-319 interfacing with ICDS API 360. Moreover, as stated above, ICDS node 430 (in conjunction with nodes 410, 420 and 440) may be configured to cache dynamic as well as static Web-based content using various distributed database algorithms.

One specific example of a data service provided by one embodiment of the present system is Wide Area Load Balancing (hereinafter "WALB") using layer 7 switching as described in the co-pending U.S. Patent Application entitled "WIDE AREA LOAD BALANCING" (Serial No. _____), which is assigned to the assignee of the present application and which is incorporated herein by reference. The present system may also perform dynamic protocol selection, dynamic query resolution, and/or heuristic adaptation for replicating content across a network as set forth in the co-pending U.S. Patent Application entitled Dynamic Protocol Selection and "QUERY RESOLUTION FOR CACHE SERVERS" (Serial No. ____), which is assigned to the assignee of the present application and which is incorporated herein by reference. Finally, the present system also may include network news (e.g., Usenet news) services set forth in the co-pending U.S. Patent Applications entitled "HYBRID NEWS SERVER" (Serial No. ____), and "SELF-MODERATED VIRTUAL COMMUNITIES" (Serial No. ____), each assigned to the assignee of the present application and each incorporated herein by reference.

Access Services

As stated above, prior art proxy server cache systems such as proxy server 210 are only capable of caching static, publicly available Web pages. A substantial amount of Web-based and non-Web-based content, however, requires some level of authentication before a user will be permitted to download it. Thus, client 472 (in Figure 4) may pay a service fee to obtain access to content on a particular web site (e.g., from server 460 or from another server over network 490). As a result, when he attempts to access content on the site he will initially be prompted to enter a user name and password. Once the user transmits this information to the Web server, he will then be permitted to download Web server content as per his service agreement.

A problem that arises, however, is that prior art cache systems such as proxy server 210 are not permitted to cache the requested content. This is because proxy server 210 has no way of authenticating subsequent users who may attempt to download the content. Thus, documents which require authentication are simply uncacheable using current network cache systems.

The present ICDS system, however, includes user authentication support embedded in access services module 322. Thus, when client 473, for example, attempts to access a Web page or other information which requires authentication, ICDS node will determine whether the requested content is stored locally. If it is, then ICDS node 430 may communicate with the authentication server (e.g., server 460 or any server that is capable of authenticating client 473's request) to determine whether client 473 should be granted access to the content. This may be accomplished using standardized authentication service definitions embedded in access services module 322. Using these definitions, ICDS node 430 will not only know what authentication server to use, it will also know what authentication *protocol* to use when it communicates to the authentication server. As a result of providing local access services module 322 for authentication, network information which requires authentication can now be cached locally in ICDS node 430, thereby conserving additional bandwidth across network link 405 and/or ISP network 480.

One particular embodiment of the present system replicates Remote Authentication Dial In User Service (hereinafter "RADIUS") information across network 480. RADIUS is an application-level protocol used by numerous ISP's to provide user authentication and profile services. This is achieved by setting up a central RADIUS server with a database of users, which provides both authentication services (i.e., verification of user name and password) and profile services detailing the type of service provided to the user (for example, SLIP, PPP, telnet, rlogin).

Users connect to one or more Network Access Servers (hereinafter "NASs") which operate as a RADIUS clients and communicate with the central RADIUS server. The NAS client passes the necessary user information to the central RADIUS server, and then acts on the response which is returned. RADIUS servers receive user connection requests, authenticate users, and then return all configuration information necessary for the client to deliver service to the user.

One problem associated with the RADIUS protocol is that it does not provide any built in facilities for replication of RADIUS information. Accordingly, on large ISP's such as America Online ("AOL"), which may have tens of millions of users, RADIUS servers are hard hit, potentially handling thousands of logon requests a minute. This may create severe performance/bandwidth problems during high traffic periods. In response, some ISP's have taken a brute-force approach to distributing RADIUS information by simply copying the information to additional servers across the network without any built in mechanism to keep the RADIUS data coherent and up-to-date.

One embodiment of the present ICDS system provides an efficient, dynamic mechanism for distributing RADIUS information. Specifically, a RADIUS module is configured to interface with ICDS API 360 in this embodiment (similar to the way in which protocol modules 310-319 interface with the ICDS API 360). RADIUS information can then be seamlessly distributed across the system using distributed database engine 361. For example, the RADIUS module in conjunction with access services module 322 on ICDS node 430 may maintain radius information for local users. [Exactly how will this work? I assume that access services module will be used but there will be a separate RADIUS protocol module to support the protocol??] Thus, when client 472 first logs in to the system, ICDS node 430 may communicate with a second ICDS node (e.g., central ICDS server 460) which contains the necessary RADIUS authentication and user profile information. Client 472 will input a user name and password and will then be permitted access to the network as per his service agreement with ISP 400.

Unlike previous RADIUS systems, however, ICDS node 430 in the present embodiment may locally cache client 472's RADIUS information so that the next time client 472 attempts to login to the network, the information will be readily available (i.e., no access to a second ICDS node will be necessary). ICDS node 430 may be configured to save client 472's RADIUS information locally for a predetermined period of time. For example, the information may be deleted if client 472 has not logged in to local ICDS node 430 for over a month.

Thus, if client 472 represents a user who frequently travels across the country and logs in to ISP 400's network 480 from various different ICDS nodes, the present system provides a quick, effective mechanism for dynamically replicating client 472's user information into

those geographical locations from which he most commonly accesses ISP 400. This reduces the load which would otherwise be borne by a central RADIUS server and also improves client 472's user experience significantly (i.e., by providing him with a quick login).

Database replication can also be used to update RADIUS information distributed across multiple ICDS nodes 410-440. This may be done using known store procedures defined in relational database 361. For example, if client 472 cancels his service agreement with ISP 400, he should not be able to continually log in to local ICDS node 430 using the RADIUS information which has been cached locally. Thus, under the present ICDS system, ISP 400 may simply issue a relational database query such as [let's add another update query here using database terminology as an example] to immediately update ICDS node 430's radius information.

One of ordinary skill in the art will readily recognize from the preceding discussion that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the invention. Throughout the foregoing description, specific embodiments of the ICDS system were described using the RADIUS protocol in order to provide a thorough understanding of the operation of the ICDS system. It will be appreciated by one having ordinary skill in the art, however, that the present invention may be practiced without such specific details. For example, the ICDS system may also distribute authentication and user profile information in the form of the Lightweight Directory Access Protocol ("LDAP"). In other instances, well known software and hardware configurations/techniques have not been described in detail in order to avoid obscuring the subject matter of the present invention.

Access services module 322 may also provide local encryption/decryption and watermarking of internet content. Audio or video content delivery systems, for example, commonly use encryption of content to protect the rights of the underlying copyright holder. When a user requests a particular piece of content some delivery systems encrypt the content using a unique client encryption key. Only a client who possesses the encryption key (presumably the client who paid for the content) will be permitted to play the content back. Other systems provide for the "watermarking" of content (rather than encrypting) so that the rightful owner of the content may be identified. This simply entails embedding a unique "tag"

which identifies the source of the content and/or the owner of the content (i.e., the one who paid for it).

Prior art caching systems such as proxy server 210 are not capable of dealing with encrypted or watermarked content because the encryption/watermarking functionality was not provided locally (i.e., proxy server 210 was not "smart" enough). In one embodiment of the present ICDS system, however, access services module 322 of ICDS nodes 410-440 includes a local encryption module and/or a local watermarking module. For example, if client 473 requests specific content such as a copyrighted music content stored on a music server (e.g., ICDS server 460), the initial request for the content will be made from ICDS node 430 on behalf of client 473. ICDS node 430 will retrieve the requested content and cache it locally. If the requested content requires encryption, ICDS node 430 will use its local encryption module to encrypt the requested content using a unique user encryption key for client 473.

If a second client – e.g., client 472 – requests the same content, the copy stored locally on ICDS module 430 can be used. ICDS module 430 will simply encrypt the content using a *different* encryption key for user 472. Thus, frequently requested multimedia content (which, as is known in the art, can occupy a substantial amount of storage space) may be cached locally at ICDS node 430 notwithstanding the fact that the content requires both user authentication and encryption.

The same functionality may be provided for watermarking of content. ICDS node will use a watermarking module (which may comprise a component of access services module 322) to individually watermark multimedia information requested by individual clients, thereby protecting the rights of the copyright holder of the underlying multimedia content. This information can then be regularly communicated back to a central database repository.

As is known in the art, multimedia files can be extremely large and, accordingly, take substantially more time to communicate across a network than do, for example, generic Web pages. As such, the ability to locally cache multimedia files significantly reduces traffic across network 480, and also significantly improves the user experience for local users when downloading multimedia information.

Transaction Services

In addition to replicating data services and access services information across a network, the present ICDS system also provides for the replication of transaction services. Transaction services includes maintaining information on client payments for use of ISP 400's services as well as information relating to the client's online access profile (i.e., recording of the times when the user is online).

When a client logs in to an ISP today, the client's online information is maintained on a single central server. The central server maintains records of when and for how long the client logged in to the network and may also include information about what the client did while he was online. As was the case with maintaining a central RADIUS server, maintaining a central transaction server for all users of a large ISP is inefficient and cumbersome. The present system solves the performance and bandwidth problems associated with such a configuration by storing transaction information locally via transaction module 323 and algorithms build around distributed database engine 361.

Thus, if client 472 only logs on to ISP 400's network 480 via ICDS node 430, all of his transaction and billing information will be stored locally. The information may then be communicated across network 480 to a central billing server at predetermined periods of time (e.g., once a month). [We didn't go into great detail on transaction services and the rest; please add information as you feel appropriate]

Commercial Services

Commercial services module 324 provides a significantly improved local caching capability for add rotation and accounting. An add rotation system operating in conjunction with a typical proxy cache server will now be described with respect to **Figure 2**. When client 120 downloads a web page from Web server 142 the Web page may contain an ad tag or an add tag may automatically be inserted. The add tag will identify add server 170 and will indicate that an add should be inserted into the requested Web page from add server 170. Add server will then identify a specific add to insert into the downloaded Web page from add content server. The Web page plus the inserted add will then be forwarded to proxy server 210 and on to client 120.

Add server 170 will keep an accounting of how many different users have downloaded Web pages with adds inserted as described above. However, one problem with accounting on this system is that proxy server 210 requests Web pages *on behalf of* its clients. Accordingly, once the requested Web pages has been cached locally in Web cache 220, add server will only receive requests from proxy server 210 for any subsequent requests for the Web page. This will result in an inaccurate accounting of how many unique clients actually requested the Web page (and how many adds were viewed by unique users).

Because one embodiment of the present system provides built in caching support for ad rotation services, an accurate accounting of the number of hits that a particular ad receives may be maintained. More specifically, one embodiment of the present ICDS system solves this problem by providing a commercial services module that monitors and records how often individual clients request Web pages containing particular adds from add content server 171. This information than then be communicated to a central server (e.g., ICDS server 460) at predetermined intervals for generating add rotation usage reports.

Directory Services

Directory services provide the ability to cache locally a directory of information across network 480 or 490. That is, the question here is not whether the particular information is available but where exactly over networks 480 or 490 it is located. It should be noted that there may be some overlap between the directory services concept and the index replication concept described above with respect to data services. [I'm still not 100% sure what this is -- please elaborate with an example]

II. Content Routing

The term "content routing" refers to the ability to select among various techniques/protocols for maintaining a coherent set of content across a network. The selection of a particular technique may be based on several routing variables including, but not limited to, the type of content involved (i.e., FTP, HTTP . . . etc), the size of the content involved (i.e., small files such as HTTP vs. large files such as audio/video streaming), the location of the content on network 480 and/or network 490, the importance of a particular piece of content (i.e., how important it is that the content be kept up-to-date across the entire network), the particular user requesting the content and the terms of his subscription agreement (i.e., some users may be willing to pay more to be insured that they receive only the most up-to-date

content without having to wait), the frequency with which the content is accessed (e.g., 5%-10% of content on the Internet represents 90% of all the *requested* content), and the underlying costs and bandwidth constraints associated with maintaining up-to-date, coherent content across a particular network (e.g., network 480).

Three content routing techniques which may be selected (based on one or more of the foregoing variables) to maintain coherent content across the plurality of nodes illustrated in **Figure 4** are content revalidation, content notification, and content synchronization.

Content Revalidation

When content validation is selected, the original content source will be checked only when the content is requested locally. For example, client 473 may request an installation program for a new Web browser (e.g., the latest version of Microsoft's™ Internet Explorer™). The file may then be transmitted from ICDS server 460 to client 473 and a copy of the file cached locally on ICDS node 430. Consequently, if client 472 requests the same program, for example, two weeks later, ICDS node 430 may be configured to check ICDS server 460 to ensure that it contains the most recent copy of the file before passing it on to client 472 (i.e., ICDS node 430 "revalidates" the copy it has locally).

ICDS node 430 may also be configured to revalidate a piece of content only if has been stored locally for a predetermined amount of time (e.g., 1 week). The particular length of time selected may be based on one or more of the variables discussed above. Moreover, in one embodiment, the age/revision of a particular piece of content is determined based on tags (e.g., HTML metatags) inserted in the particular content/file.

Revalidation may work more efficiently with certain types of content than with others. For example, revalidation may be an appropriate mechanism for maintaining up-to-date copies of larger files which do not change very frequently (i.e., such as the program installation files described above). However, revalidation may not work as efficiently for caching smaller and/or continually changing files (e.g., small HTML files) because the step of revalidating may be just as time consuming as making a direct request to ICDS server 460 for the file itself. If the file in question is relatively small and/or is changing on a minute-by-minute basis (e.g., an HTML file containing stock quotes) then one or more other content routing techniques may be more appropriate.

Of course, other routing variables may influence the decision on which technique to use, including the issue of how strong the data transmission connection is between ICDS node 430 and ICDS server 460 (i.e., how reliable it is, how much bandwidth is available . . . etc) and the necessity that the underlying information cached locally (at ICDS node 430) be accurate. The important thing to remember is that ICDS node 430 – because of its underlying open API architecture – may be configured based on the unique preferences of a particular client.

Content Notification

Content notification is a mechanism wherein the central repository for a particular piece of content maintains a list of nodes, or “subscribers,” which cache a copy of the content locally. For example, in Figure 4, a plurality of agents may run on ICDS server 460 which maintain a list of content subscribers (e.g., ICDS node 430, ICDS node 420 . . . etc) for specific types of content (e.g., HTML, data streaming files, FTP files . . . etc). In one embodiment of the system, a different agent may be executed for each protocol supported by ICDS server 460 and/or ICDS nodes 410–440.

When a particular piece of content is modified on ICDS server 460, a notification of the modification may be sent to all subscriber nodes (i.e., nodes which subscribe to that particular content). Upon receiving the notification, the subscriber node – e.g., ICDS node 430 – may then invalidate the copy of the content which it is storing locally. Accordingly, the next time the content is requested by a client (e.g., client 472), ICDS node 430 will retrieve the up-to-date copy of the content from ICDS server 460. The new copy will then be maintained locally on ICDS node 430 until ICDS node 430 receives a second notification from an agent running on ICDS server 460 indicating that a new copy exists.

Alternatively, each time content is modified on ICDS server 460 the modified content may be sent to all subscriber nodes along with the notification. In this manner a local, up-to-date copy of the content is always ensured. In one embodiment of the system, notification and/or transmittal of the updated content by the various system agents is done after a predetermined period of time has elapsed (e.g., update twice a day). The time period may be selected based on the importance of having an up-to-date copy across all nodes on the network 480, 490.

As was the case with content revalidation, the different varieties of content notification may work more efficiently in some situations than in others. Accordingly, content notification may be selected as a protocol (or not selected) based on one or more of the routing variables recited at the beginning of this section (i.e., the "content routing" section). For example, content notification may be an appropriate technique for content which is frequently requested at the various nodes across networks 480 and 490 (e.g., for the 5-10% of the content which is requested 90% of the time), but may be a less practical technique for larger amount of content which is requested infrequently. As another example, large files which change frequently may not be well suited for content notification (i.e., particularly the type of content notification where the actual file is sent to all subscribers along with the notification) due to bandwidth constraints across networks 480 and/or 490 (i.e., the continuous transmission of large, frequently changing files may create too much additional network traffic).

Content Synchronization

Content synchronization is a technique for maintaining an exact copy of a particular type of content on all nodes on which it is stored. Using content synchronization, as soon as a particular piece of content is modified at, for example, ICDS node 430, it will immediately be updated at all other nodes across networks 480 and/or 490. If the same piece of data was concurrently modified at one of its other storage locations (e.g., ICDS node 410) then the changes may be backed off in order to maintain data coherency. Alternatively, an attempt may be made to reconcile the two separate modifications if it is possible to do so (using, e.g., various data coherency techniques).

Once again, as with content notification and content revalidation, content synchronization is more suitable for some situations than it is for others. For example, content synchronization is particularly useful for information which can be modified from several different network nodes (by contrast, the typical content notification paradigm assumes that the content is modified at one central node). Moreover, content synchronization may be useful for maintaining content across a network which it is particularly important to keep current. For example, if network 480 is an automatic teller machine (hereinafter "ATM") network, then when a user withdraws cash from a first node (e.g., ICDS node 440), his account will be instantly updated on all nodes (e.g., ICDS node 410, 420, 460, and 430) to reflect the withdrawal. Accordingly, the user would not be able to go to a different node in a different part of the country and withdraw more than what he actually has in his account.

As another example, a user's account status on a network (i.e., whether he is a current subscriber and/or what his network privileges are) may be maintained using content synchronization. If, for example, a user of network 480 were arrested for breaking the law over network 480 (e.g., distributing child pornography), it would be important to disable his user account on all network nodes on which this information might be cached. Accordingly, using content synchronization, once his account was disabled at one node on network 480 this change would automatically be reflected across all nodes on the network.

As previously stated, the choice of which content routing technique to use for a particular type of content may be based on any of the variables set forth above. In one embodiment, the frequency with which content is accessed across the networks 480 and/or 490 may be an important factor in deciding which protocol to use. For example, the top 1% accessed content may be selected for content synchronization, the top 2%-10% accessed content may be selected for content notification, and the remaining content across networks 480, 490 may be selected for content revaidation.

III. Content Delivery Medium Selection

In addition to the content routing flexibility provided by the content delivery system as set forth above, one embodiment of the system allows content delivery nodes such as ICDS node 430 to select from a plurality of different transmission media. For example, ICDS node 430 may receive content from ICDS server 460 via a plurality of communication media, including, but not limited to, satellite transmission, wireless RF transmission, and terrestrial transmission (e.g., fiber).

Moreover, as with the selection of a particular content routing technique, the selection of a particular transmission medium may be based on any of the variables set forth above (see, e.g., routing variables listed under counter routing heading; page 24, line 18 through page 25, line 9). Moreover, the choice of a particular transmission medium may be dynamically adjustable based on performance of that medium. For example, ICDS node 430 may be configured to receive all of its content over terrestrial network 480 as long as network 480 is transmitting content at or above a threshold bandwidth. When transmissions over network 480 dip below the threshold bandwidth, ICDS node 480 may then begin receiving certain content via satellite broadcast or wireless communication.

In addition, a transmission medium may be selected for transmitting specific content based on how frequently that content is accessed. For example, the top 10% frequently accessed content may be continually pushed out to ICDS node 430 via satellite broadcast while the remaining content may be retrieved by (i.e., "pulled" to) ICDS node 430 over network 480 upon request by clients (e.g., client 473). Accordingly, those employing ICDS nodes such as node 430 can run a cost-benefit analysis to determine the most cost effective way to implement their system by taking in to consideration, for example, the needs of their users, the importance of the content involved and the expense of maintaining multiple transmission connections into ICDS node 430 (e.g., the cost associated with maintaining an ongoing satellite connection).

In one embodiment of the system, tags (e.g., HTML metatags) may be inserted into particular types of content to identify a specific transmission path/medium for delivering that content to ICDS node 480. The tags in this embodiment may identify to various nodes (and/or routers) across networks 480 and/or 490 how the particular content should be routed across the networks (e.g., from node 410 to node 420 via terrestrial network 480; from node 420 to node 430 via wireless transmission).

One of ordinary skill in the art will readily recognize from the preceding discussion that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the invention. Throughout this detailed description, numerous specific details are set forth such as specific network protocols (i.e., RADIUS) and networks (i.e., the Internet) in order to provide a thorough understanding of the present invention. It will be appreciated by one having ordinary skill in the art, however, that the present invention may be practiced without such specific details. In other instances, well known software and hardware configurations/techniques have not been described in detail in order to avoid obscuring the subject matter of the present invention. The invention should, therefore, be measured in terms of the claims which follow.

CLAIMS

What is claimed is:

1. A network content delivery system configured to:
select a first content routing technique for processing a first set of network content;
and
select a second content routing technique for processing a second set of network content, wherein said first and second content routing techniques are selected based on one or more content routing variables.
2. The network content delivery system as claimed in Claim 1 wherein one of said selected content routing techniques is a content revalidation technique.
3. The network content delivery system as claimed in Claim 1 wherein one of said selected content routing techniques is a content notification technique.
4. The network content delivery system as claimed in Claim 1 wherein one of said selected content routing techniques is a content synchronization technique.
5. The network content delivery system as claimed in Claim 1 wherein one of said content routing variables is the frequency with which said network content is accessed by users.
6. The network content delivery system as claimed in Claim 1 wherein one of said content routing variables is the size of said network content.
7. The network content delivery system as claimed in Claim 1 wherein one of said content routing variables is the frequency with which said network content is modified.
8. The network content delivery system as claimed in Claim 1 wherein one of said content routing variables is the type of network content (e.g., HTML, Usenet News).
9. The network content delivery system as claimed in Claim 1 wherein one of said content routing variables is identity of the user requesting said network content.
10. The network content delivery system as claimed in Claim 2 wherein said content revalidation technique is selected based on the size of said network content.
11. The network content delivery system as claimed in Claim 2 wherein said content revalidation technique is selected based on the frequency with which said network content is accessed.
12. The network content delivery system as claimed in Claim 3 wherein said content notification technique is selected based on the size of said network content.

13. The network content delivery system as claimed in Claim 3 wherein said content notification technique is selected based on the frequency with which said network content is accessed.

14. The network content delivery system as claimed in Claim 4 wherein said content synchronization technique is selected based on the size of said network content.

15. The network content delivery system as claimed in Claim 4 wherein said content synchronization technique is selected based on the frequency with which said network content is accessed.

16. The network content delivery system as claimed in Claim 1 including the additional step of selecting a first transmission medium for a first group of network content based on one or more of said content routing variables.

17. The network content delivery system as claimed in Claim 16 including the additional step of selecting a second transmission medium for a second group of network content based on one or more of said content routing variables.

18. The network content delivery system as claimed in Claim 1 including an application programming interface for interfacing with a plurality of network protocol and service modules.

19. A content delivery system comprising:
a network node for storing network content;
a first transmission medium communicatively coupled to said network node for transmitting a first set of network content to said network node; and
a second transmission medium communicatively coupled to said network node for transmitting a second set of network content to said network node,
wherein said first and second sets of network content are selected based on one or more routing variables.

20. The content delivery system as claimed in Claim 19 wherein said first transmission medium is a satellite transmission.

21. The content delivery system as claimed in Claim 19 wherein said first transmission medium is a wireless radio frequency transmission.

22. The content delivery system as claimed in Claim 19 wherein said first transmission medium is terrestrial-based transmission.

23. The content delivery system as claimed in Claim 19 wherein said network node monitors transmission bandwidth of said first transmission medium and reallocates content

from said first set to said second set if said first transmission medium drops below a predetermined threshold value.

24. The content delivery system as claimed in Claim 23 wherein said first transmission medium is terrestrial and said second transmission medium is non-terrestrial.

25. The content delivery system as claimed in Claim 19 wherein content is included in said first set based on the frequency with which said content is accessed.

26. The network content delivery system as claimed in Claim 19 including an application programming interface for interfacing with a plurality of network protocol and service modules.

27. An article of manufacture including a sequence of instructions stored on a computer-readable media which, when executed by a network node, cause the network node to perform the acts of:

establishing a plurality of groups of network content to be cached on said network node based on one or more content routing variables;

selecting a first content routing technique for maintaining data coherency in a first group of said plurality; and

selecting a second content routing technique for maintaining data coherency in a second group of said plurality.

28. The article of manufacture as claimed in claim 27 wherein said first content routing technique is content revalidation.

29. The article of manufacture as claimed in Claim 28 wherein said second content routing technique is content notification.

30. The article of manufacture as claimed in Claim 28 wherein said second content routing technique is content synchronization.

31. The article of manufacture as claimed in Claim 28 wherein said content routing variable used to select said content for said first group is the frequency with which said content is accessed.

32. The article of manufacture as claimed in Claim 29 wherein said content routing variable used to select said content for said first group is the frequency with which said content is accessed.

33. The article of manufacture as claimed in Claim 30 wherein said content routing variable used to select said content for said first group is the frequency with which said content is accessed.

34. A network node comprising:

an application programming interface ("API"), said API including a distributed relational database engine;

a plurality of protocol modules for interfacing with said API, said protocol modules configured to allow said system to communicate over a network using a plurality of network protocols;

a cache memory for caching data communicated to said cache memory using said plurality of protocol modules; and

a data services module for maintaining coherency between said data stored in said cache memory and data stored at other nodes across said network.

1/4

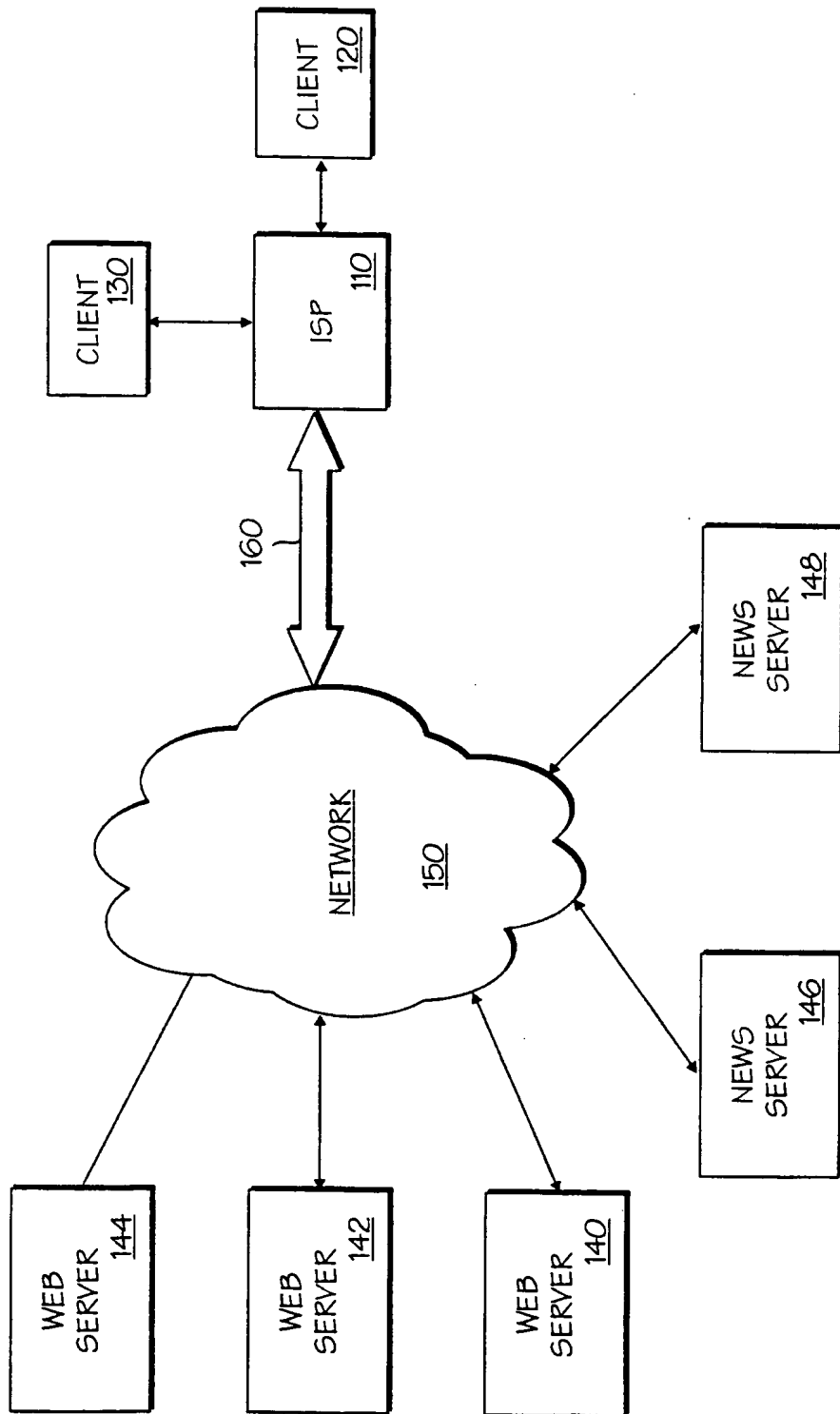


FIG. 1

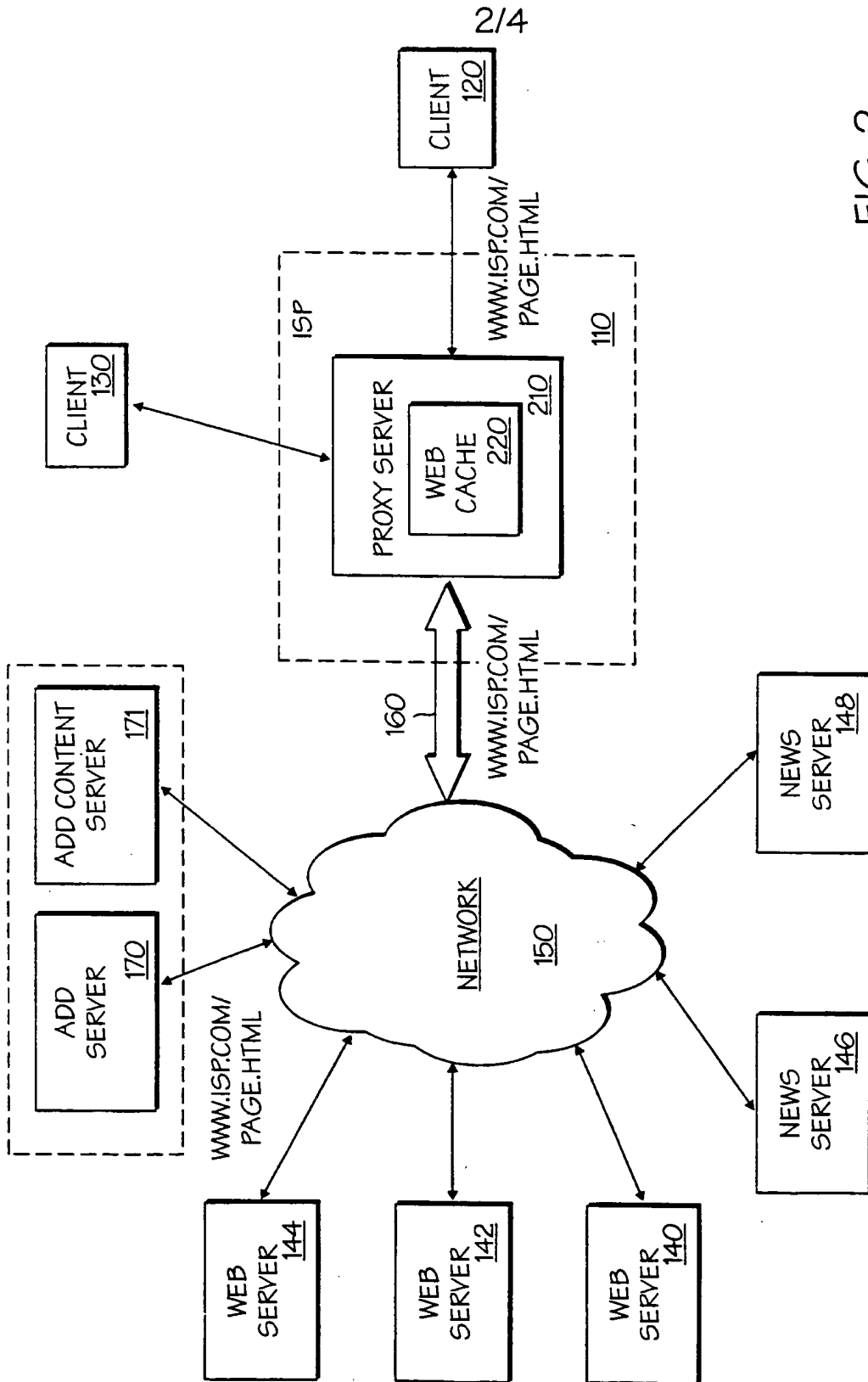


FIG. 2

3/4

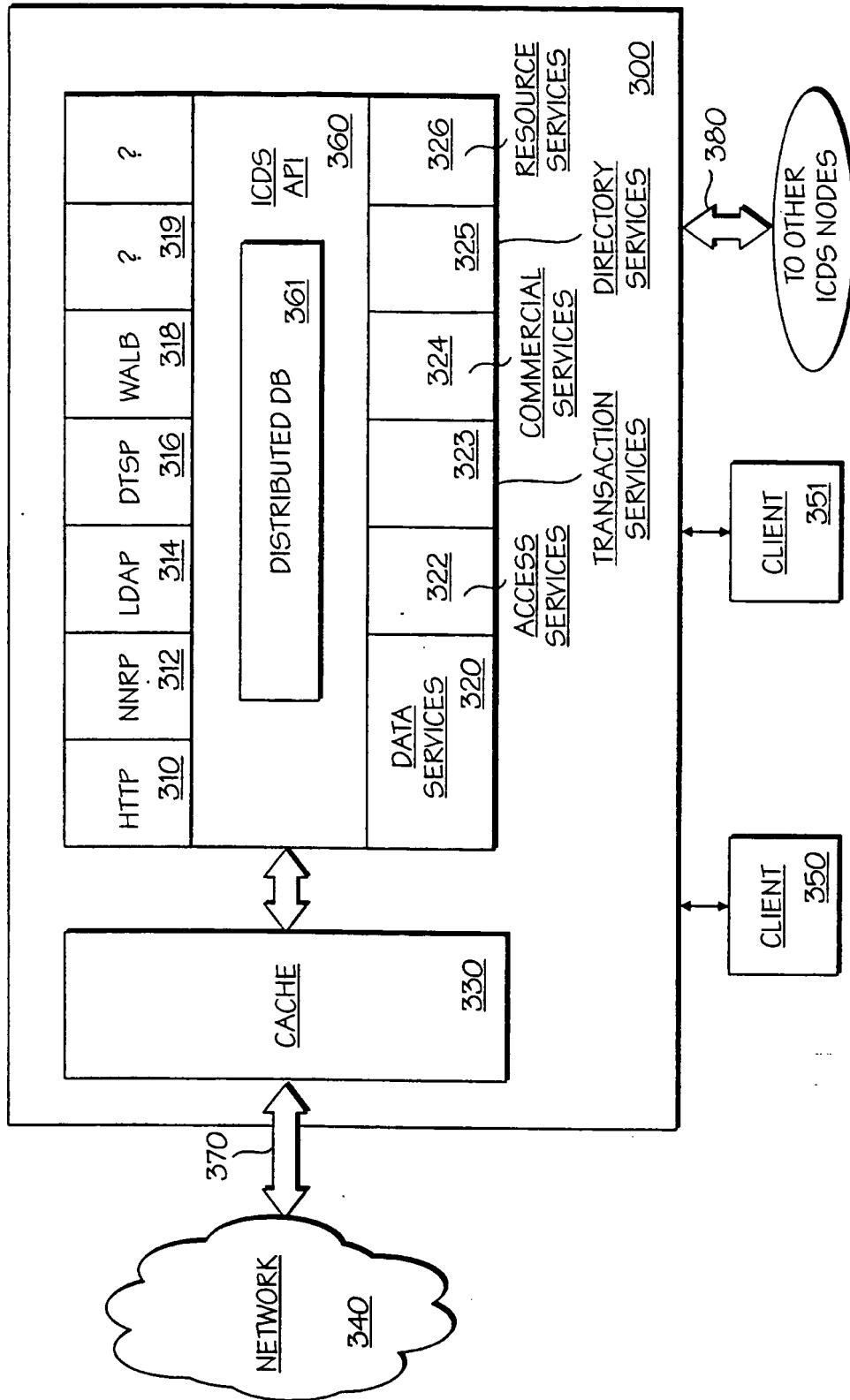


FIG. 3

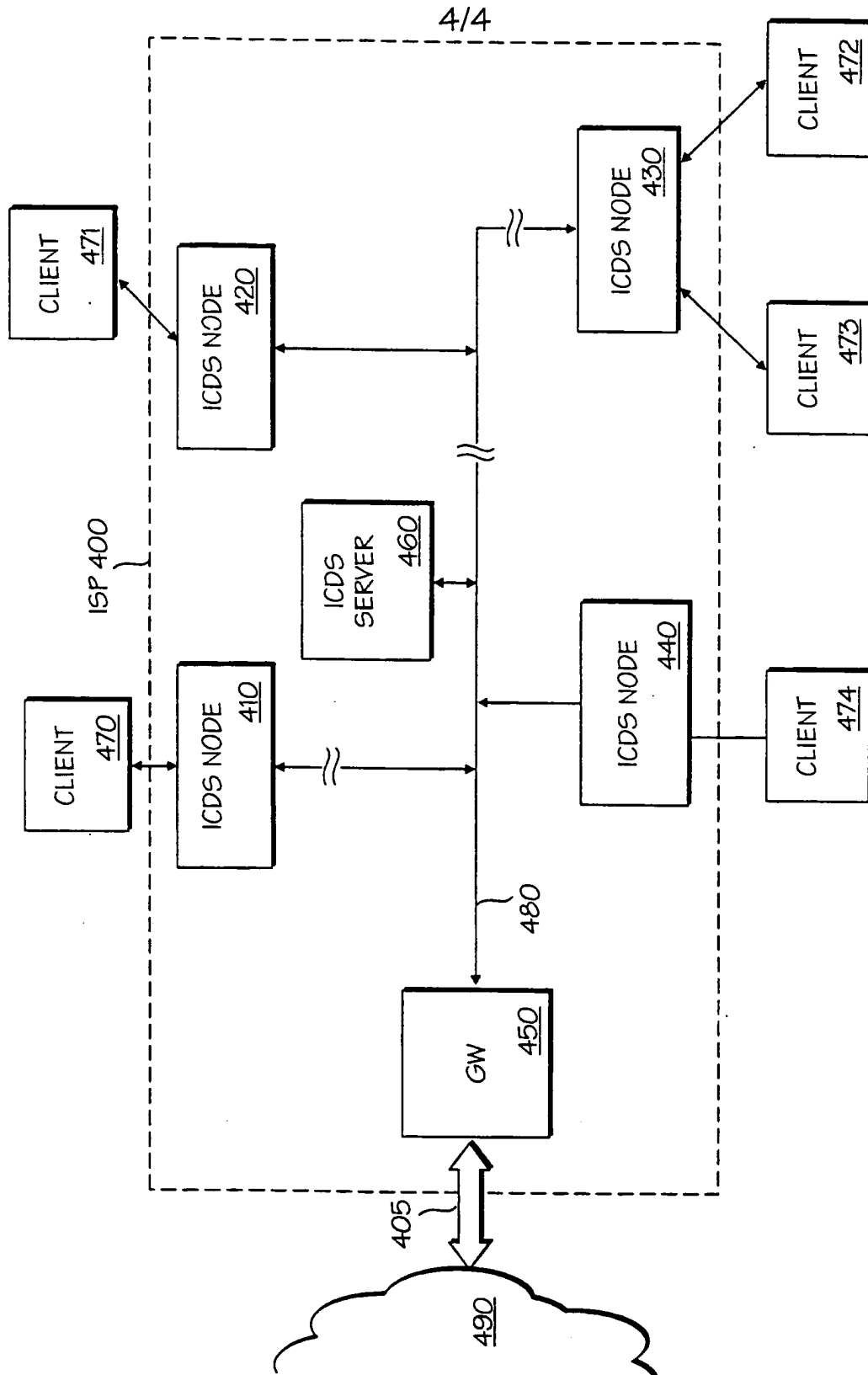


FIG. 4



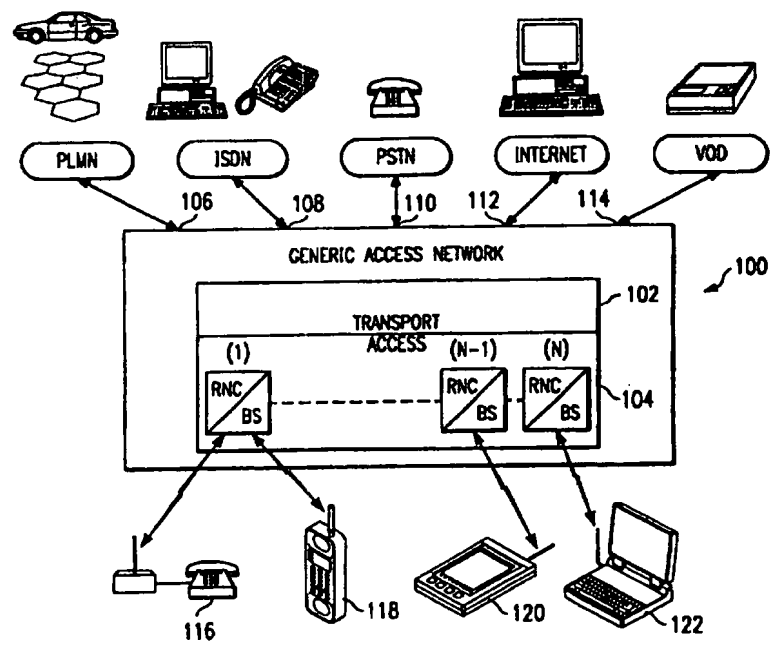
INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<p>(51) International Patent Classification ⁶ : H04L 9/08, H04Q 7/32</p>	<p>A1</p>	<p>(11) International Publication Number: WO 98/10561 (43) International Publication Date: 12 March 1998 (12.03.98)</p>
<p>(21) International Application Number: PCT/SE97/01407 (22) International Filing Date: 26 August 1997 (26.08.97) (30) Priority Data: 08/708,796 9 September 1996 (09.09.96) US (71) Applicant: TELEFONAKTIEBOLAGET LM ERICSSON (publ) [SE/SE]; S-126 25 Stockholm (SE). (72) Inventor: RUNE, Johan; Motionsvägen 5, S-181 30 Lidingö (SE). (74) Agents: BANDELIN, Hans et al.; Telefonaktiebolaget LM Ericsson, Patent and Trademark Dept., S-126 25 Stockholm (SE).</p>	<p>(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).</p> <p>Published With international search report.</p>	

(54) Title: METHOD AND APPARATUS FOR ENCRYPTING RADIO TRAFFIC IN A TELECOMMUNICATIONS NETWORK

(57) Abstract

A generic communications network (100) provides an encrypted communications interface between service networks (130, 132, 134) and their subscribers. When communications are initiated between a subscribing communications terminal (118) and the generic network (100), the terminal (118) compares a stored network identifier associated with a stored public key, with a unique identifier broadcast by the generic network (100). If a match is found, the terminal (118) generates a random secret key, encrypts the secret key with the stored public key, and transmits the encrypted secret key. The generic communications network (100) decrypts the secret key using a private key associated with the public key. The secret key is used thereafter by the terminal (118) and the generic network (100) to encrypt and decrypt the ensuing radio traffic. Consequently, the network (100) can maintain secure communications with the terminal (118) without ever knowing the terminal's identity.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakistan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

-1-

**METHOD AND APPARATUS FOR ENCRYPTING RADIO TRAFFIC
IN A TELECOMMUNICATIONS NETWORK**

BACKGROUND OF THE INVENTION

5 Technical Field of the Invention

The present invention relates generally to the field of wireless radio communications and, in particular, to a method and apparatus for encrypting radio traffic between terminals and a mobile communications network.

Description of Related Art

10 The need for increased mobility and versatility in telecommunications networks requires the networks to cover larger geographical areas and provide a broader range of telecommunications services to subscribers. These telecommunications services include teleservices and bearer services. A teleservice provides the necessary hardware and software for a subscriber to communicate with
15 another subscriber (e.g., terminal, etc.). A bearer service provides the capacity required to transmit appropriate signals between two access points (e.g., ports) that provide an interface with a network. Telecommunications services can be provided to subscribers by a number of service networks, such as, for example, public land mobile telecommunications networks (PLMNs), public switched telephone networks
20 (PSTNs), integrated services digital networks (ISDNs), the so-called "Internet" access networks, video on demand (VOD) networks, and other proprietary service networks.

 In response to the need for increased mobility and versatility, a new mobile radio telecommunications network is being developed, which has a generic interface
25 through which a service network subscriber can be connected with that service network regardless of the subscriber's geographic location. This generic mobile radio network is referred to as the "Generic Access Network" (GAN). In order to more readily understand the present invention, which deals primarily with encrypting communications traffic between terminals and a GAN, a brief description
30 of such a GAN is provided below with respect to FIGURE 1.

-2-

FIGURE 1 is a perspective view of an exemplary GAN connected to a plurality of service networks and service network subscribers. The GAN (10) illustrated by FIGURE 1 includes an access network interconnected with a transport network. The access network includes a plurality of base stations (e.g., BS1 and BS2). Each base station includes a radio transmitter and receiver that provides communications coverage for a respective geographical area (e.g., a so-called cell, C1 and C2). The base stations are connected to a radio network controller (RNC) 12. Although not shown explicitly, certain of the base stations can be connected to RNC 12 (e.g., BS1 and BS2), and certain other of the base stations can be connected to one or more other RNCs. A plurality of the RNCs can be interconnected to provide a communications path therebetween. The RNCs distribute signals to and from the connected base stations.

A plurality of service networks (e.g., VOD network, PLMN, PSTN, Internet) are connected through respective access input ports (14, 16, 18, 20, 22, 24 and 26) to the access network of GAN 10. Each service network uses its own standard signaling protocol to communicate between its internal signaling nodes. For example, the Global System for Mobile communications (GSM), which is a digital cellular PLMN that has been fielded throughout Europe, uses the Multiple Application Part (MAP) signaling protocol. As illustrated by FIGURE 1, the RNCs in the access network are connected through at least one of the access input ports to a service network. As shown, RNC 12 is connected through access ports 20 and 24, respectively, to the PLMN and PSTN service networks.

Mobile terminals 28 and 30 are located within the radio coverage area of GAN 10, and can establish a connection with each of the base stations (e.g., BS2) in the access network. These mobile terminals can be, for example, a cellular phone, mobile radiotelephone, personal computer (notebook, laptop, etc.) possibly connected to a digital cellular phone, or mobile television receiver (for VOD). Signal transport between a mobile terminal and a selected service network takes place over specified signal carriers. For example, signals are transported between the cellular phone (28) and the PLMN service network over signal carriers SC1 and SC2.

-3-

The mobile terminals (e.g., 28 and 30) include an access section and service network section. The access section of a mobile terminal is a logical part of the access network and handles the signaling required to establish the signal carrier (e.g., SC2 and SC4) between the mobile terminals and RNC 12. The service network section of a mobile terminal is a logical part of the service network to which that terminal's user subscribes. The service network section of a mobile terminal receives and transmits signals, in accordance with the specified standards of its related service network, via the established signal carriers SC1 and SC2 (or SC4). The radio interface portion of the signal carrier SC2 or SC4 (between the mobile terminal and base station) can be time division multiple access (TDMA), code division multiple access (CDMA), or any other type of multiple access interface.

The service network subscribers can access their respective service network through the GAN. The GAN provides a signal carrier interface that allows a message to be transported transparently over a signal carrier (e.g., SC1 and SC2) between the service network part of a mobile terminal and its service network. The GAN accomplishes this function by matching the characteristics of the signaling connections and traffic connections of all of the service networks that connect to it. Consequently, the GAN can extend the coverage of existing service networks and also increase the subscribers' degree of mobility.

A unique characteristic of a GAN is that it has no subscribers of its own. The mobile users of the GAN are permanent subscribers to their own service networks, but they are only temporary users of the GAN. Consequently, a GAN does not know (or need to know) the identity of these users. However, a problem arises in attempting to encrypt radio traffic between the mobile terminals and the GAN.

Radio traffic (e.g., speech information or data) between mobile terminals and base stations is typically encrypted to ensure that the information being passed remains confidential. Although some service networks (e.g., GSM) encrypt traffic, most other service networks do not. Consequently, a GAN should be capable of encrypting traffic for those service networks that do not have that capability.

-4-

However, since a GAN does not know the identity of its users (the service network subscribers), it must be capable of encrypting radio traffic using encryption keys that are created without knowing a subscribing terminal's identity or authenticity. Unfortunately, most existing mobile communications networks use encryption techniques that generate encryption keys by using authentication parameters. In other words, to encrypt radio traffic in a conventional mobile communications network, the user terminal's identity must be known.

SUMMARY OF THE INVENTION

It is an object of the present invention to encrypt communications between a mobile terminal and a communications network without requiring the network to know the identity of the terminal.

It is also an object of the present invention to encrypt communications between a plurality of mobile terminals and a communications network without requiring the network to maintain individual encryption keys for each of the terminals.

It is another object of the present invention to encrypt communications between a mobile terminal and a communications network without requiring the terminal to permanently store a secret encryption key.

It is yet another object of the present invention to minimize call setup time, minimize transmission delays, and maximize data throughput, while encrypting communications between a mobile terminal and a communications network.

In accordance with one aspect of the present invention, a method is provided for encrypting communications between a communications network and a communications terminal, by storing a public key associated with the network at the terminal, generating a secret key at the terminal, encrypting the secret key with the stored public key at the terminal, transmitting the encrypted secret key from the terminal, receiving the encrypted secret key at the network, decrypting the received encrypted secret key with a private key, where the private key is associated with the public key, and encrypting the ensuing traffic with the secret key. If a public key has not been stored at the terminal, then the terminal transmits a request to the

-5-

network for a public key. As such, the network is not required to know the identity of the terminal in order to maintain encrypted communications with the terminal.

In accordance with another aspect of the present invention, the foregoing and other objects are achieved by a method and an apparatus for encrypting traffic
5 between a communications network and a communications terminal by broadcasting a (asymmetric) public key from the network. The public key is received by the terminal. The network maintains a private key that can be used to decrypt information encrypted with the public key. The terminal generates and stores a naturally occurring random number as a secret session (symmetric) key, encrypts the
10 symmetric session key with the public key, and transmits the encrypted session key to the network. The network decrypts the session key with the private key, and both the network and terminal encrypt the ensuing communications with the secret session key. Again, the communications network is not required to know the identity of the terminal in order to maintain encrypted communications with the terminal.

15

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete understanding of the method and apparatus of the present invention may be had by reference to the following detailed description when taken in conjunction with the accompanying drawings wherein:

20 FIGURE 1 is a perspective view of an exemplary generic access network connected to a plurality of service networks and service network subscribers;

FIGURE 2 is a top level schematic block diagram of a generic access network in which a method of encrypting radio traffic between service networks and service network subscribers can be implemented, in accordance with a preferred
25 embodiment of the present invention;

FIGURE 3 is a schematic block diagram of the access network illustrated in FIGURE 2;

FIGURE 4 is a sequence diagram that illustrates a method that can be used to encrypt radio communications between a generic access network and a terminal,
30 in accordance with a preferred embodiment of the present invention; and

-6-

FIGURE 5 is a block diagram of a method that can be used to certify the authenticity of a public key and the owner of the key with a digital signature, in accordance with a preferred embodiment of the present invention.

5 DETAILED DESCRIPTION OF THE DRAWINGS

The preferred embodiment of the present invention and its advantages are best understood by referring to FIGURES 1-5 of the drawings, like numerals being used for like and corresponding parts of the various drawings.

Essentially, in accordance with a preferred embodiment of the present
10 invention, a mobile terminal stores at least one public key, along with a unique identification character of at least one GAN associated with that public key, in a memory location. A GAN continuously broadcasts its unique identification character in all cells connected to that GAN. When contact is initiated between the terminal and that GAN, the terminal compares the received identifier with the stored
15 identifier(s), and if a match can be made, the terminal generates a random secret key, encrypts the secret key with the public key associated with that GAN's identifier, and transmits the encrypted secret key. The GAN decrypts the secret key using a private key associated with the public key. The secret key is used thereafter by the terminal and the GAN to encrypt and decrypt the ensuing radio traffic.
20 Notably, the GAN can maintain secure communications with the terminal without ever knowing the terminal's identity. Furthermore, since the GAN does not need to know the identity of such a terminal, the GAN is not required to maintain a database of individual terminal encryption keys. Additionally, the terminal is not required to store its own secret key, because it can generate a new secret key for
25 each communications session.

FIGURE 2 is a top level schematic block diagram of a generic access network in which a method of encrypting radio traffic between service networks and service network subscribers can be implemented, in accordance with a preferred embodiment of the present invention. A GAN 100 is shown, which includes a
30 transport network 102 interconnected with an access network 104. A plurality of service networks (e.g., PLMN, ISDN, PSTN, INTERNET, VOD) are connected

-7-

through respective access ports (e.g., 106, 108, 110, 112, 114) to transport network 102 and access network 104. Access network 104 includes a plurality of RNCs and associated base stations (e.g., RNC(1)-RNC(N)). The plurality of RNCs and associated base stations are connected by a respective radio interface to a plurality of mobile transceivers (terminals) 116, 118, 120 and 122. A user of each mobile terminal is a subscriber to at least one of the service networks PLMN, etc. The mobile terminals can communicate with their respective service networks in the manner described above with respect to FIGURE 1. More specifically, the RNCs control communications between the terminals and their respective service networks. Notably, although a plurality of mobile terminals (116, etc.) are shown in FIGURE 2, this is for illustrative purposes only. One or more fixed radio terminals may also be connected to GAN 100 and are thus capable of communicating with at least one of the service networks.

FIGURE 3 is a schematic block diagram of access network 104 illustrated in FIGURE 2. Access network 104 includes a plurality of RNCs (e.g., RNC(1)-RNC(N)). However, although a plurality of RNCs is shown for this embodiment, the present invention can be implemented with only one RNC. At least one service network (e.g., 130, 132, 134) is connected through at least one respective access port (e.g., AP1, AP(N-1), AP(N)) to at least one RNC. At least one base station (e.g., BS(1), BS(N)) is connected to a respective RNC (e.g., RNC(1), RNC(N)). Although a plurality of base stations is shown, the present invention can be implemented with only one base station.

A mobile terminal (e.g., cellular phone 118) is connected by a radio interface to base station BS(1). It should be readily understood that one terminal (118) is shown for illustrative purposes only and that one or more additional terminals could be shown. The RNCs (e.g., RNC(1)-RNC(N)) are interconnected by communications lines (136, 138) for communications therebetween. Consequently, terminal 118 can establish communications with any of the service networks (e.g., 130, 132, 134) through access network 104 and GAN 100 (FIGURE 2). Notably, the coverage provided for each service network can be enlarged by switching to a different access port of access network 104. In other words, terminal 118 can

-8-

communicate with service network 132 through RNC(1), interconnecting line 136, and RNC(N-1). Alternatively, if service network 132 is switched to access port AP(1), terminal 118 can communicate with service network 132 through RNC(1).

5
10
15
20
FIGURE 4 is a sequence diagram that illustrates a method that can be used to encrypt radio communications between a generic access network and a terminal, in accordance with a preferred embodiment of the present invention. The method 200 of encrypting communications can begin at the GAN or the terminal. For example, in this embodiment, at step 204, the GAN (e.g., 10) continuously broadcasts a unique identification character in all cells connected to that GAN. The terminal (e.g., 118) contains a non-volatile memory located in a GAN section of the terminal. The terminal stores at least one public key in the non-volatile memory. Along with each public key, the terminal also stores a respective expiration date for the key, and a GAN identification character that identifies a specific GAN associated with that key. In other words, each public key stored in the terminal's memory is thereby associated with a specific GAN. The terminal initiates contact by registering with a GAN (but not necessarily setting up a call). A processor in the terminal compares the received GAN identifier with the stored identifiers, and if a match can be made (and the key has not expired), the processor retrieves the stored public key associated with the identified GAN. However, in the event that no such match is found, the terminal sends a request for the GAN to transmit a public key. The transmitted public key (and its expiration date) is stored in the terminal and can be used to encrypt a secret key in the current and ensuing communication sessions.

25
At step 206, the terminal generates a (symmetric) secret key (described in detail below). At step 208, the terminal uses the retrieved public key to encrypt the secret key. At step 210, the terminal transmits the encrypted secret key to the identified GAN. At step 212, the GAN decrypts the secret key, which, at step 214, is used by the GAN and the terminal for encrypting traffic during the ensuing communications session (described in detail below).

30
Alternatively, at the end of a session with a GAN, the terminal stores the public key used for that session. When the terminal or a GAN begins a new communications session, the terminal retrieves the public key stored from the last

-9-

session with a GAN, and uses that public key to encrypt a secret key to be used for the ensuing session. If the use of that stored public key is unsuccessful, the terminal then sends a request to the GAN for a new public key. This technique advantageously increases network throughput, because a network channel is not tied up transmitting a public key. However, if a public key has not been stored from a past session with a particular GAN, the terminal can still receive the public key by requesting it from the GAN and using it to encrypt a secret key that will be used for the ensuing session. In any event, by storing the relatively large (bit-wise) public keys in the terminal, as opposed to transmitting them from the GAN, radio transmission delays can be reduced significantly, a substantial amount of network transmission time can be saved, and data throughput will be increased.

FIGURE 4 also illustrates a method that can be used to encrypt radio communications between a generic access network and a mobile terminal, in accordance with another embodiment of the present invention. For example, when communications are desired between a service network and a terminal (e.g., PLMN and terminal 118), the service network or terminal can initiate communications with a call setup message. At step 202, as the initial connection between the GAN and the terminal is established, the service network can request that the ensuing traffic will be encrypted. If so, at step 204, still during the initial call setup process, the terminal receives a public key which is continuously broadcast from one or more base stations (e.g., BS(1)-BS(N)).

In this embodiment, all of the RNCs maintain at least one public key/private key pair (the same pair in every RNC) in a memory storage location. The public key that was broadcast by the GAN is received by the terminal (118) that has initiated contact with that GAN. Preferably, both the call setup procedure and the procedure to transfer the public key is performed by an RNC, which is connected through an access port to the service network of interest (e.g., RNC(1) to AP(1) to PLMN 130). Alternatively, a base station (e.g., BS1) can be configured to maintain public/private key pairs and broadcast or otherwise transfer a public key to a terminal.

-10-

The RNC can broadcast the public key in all cells in the RNC's coverage area. Consequently, since the GAN broadcasts the public key instead of having the terminal request the key from the GAN, the terminal can register with the GAN much faster, and a call can be set up in a substantially shorter period of time. Alternatively, instead of broadcasting the public key in a plurality of cells, the RNC can transfer the public key directly through the base station that has established contact with the terminal. However, the method of broadcasting the public key in a plurality of cells before call setup advantageously decreases the load on the GAN's dedicated traffic channels.

For all embodiments, as long as the terminal is registered with the GAN, the same public key can be used for all subsequent communications with that GAN, because the same key is stored at the GAN and also at the terminal. Alternatively, the public key can be changed periodically in accordance with a predetermined scheme or algorithm, or even at the whim of the GAN operator. If an operator desires to change public keys periodically, storing each public key's expiration date at the terminal facilitates their use in that regard. Furthermore, in the preferred embodiment, when the public key is changed, it can be broadcast by the GAN for a predetermined period of time, to minimize the number of terminal requests for a new public key.

As described earlier, at step 202, the GAN can maintain one or more asymmetric public key/private key pairs. In that event, a so-called "RSA Algorithm" can be used to create the public key/private key pairs. The RSA Algorithm combines the difficulty of factoring a prime number with the ease of generating large prime numbers (using a probabilistic algorithm) to split an encryption key into a public part and a private part.

Specifically, assuming that the letters P and Q represent prime numbers, the letter M represents an unencrypted message, and the letter C represents the encrypted form of M, the RSA Algorithm can be expressed as follows:

$$M^E \text{ mod } PQ = > C \text{ (encrypted message M)} \quad (1)$$

$$C^D \text{ mod } PQ = > M \text{ (decrypted message C)} \quad (2)$$

-11-

where the term $(DE-1)$ is a multiple of $(P-1)(Q-1)$. In this embodiment, the exponent E is set to 3. The private and public keys are each composed of two numbers. For example, the numbers represented by (PQ, D) make up the private key, and the numbers represented by (PQ, E) make up the public key. Since the same value for E is used consistently, only the PQ portion of the number need be sent on request or broadcast and used for the public key (e.g., at step 204). By knowing the private key, any message encrypted with the public key can be decrypted.

Returning to FIGURE 4, at step 206, the terminal (118) receives and/or stores the asymmetric public key. The terminal generates a random symmetric secret key. The random secret key, which is used to encrypt communications preferably for the complete session, can be generated in at least one of four ways. Using one method, the terminal takes several samples from measurements of the strength of the received signal, concatenates the lower order bits of the several samples, and processes the result to produce a random number. Since the lower order bits of the received signal are well within the noise level of the received signal, a naturally occurring, truly random number is generated. A second random number generating method is to use the random noise signal created at the input of an A/D converter connected to a microphone. Again, using this method, a naturally occurring, truly random number can be generated for the secret key. A third random number generating method is for the terminal to take samples from phase measurements of the received signal, concatenate the lower order bits of the samples, and process the result to produce a random number. A fourth random number generating method is for the terminal to take samples from the encoding section of the speech codec, concatenate the lower order bits of the samples, and process the result to produce the random number.

Alternatively, a random number generated at the terminal can be used as a seed for a pseudorandom number generator. The seed is encrypted with the public key from the GAN, and transmitted to the GAN. The seed is used simultaneously in the GAN and the terminal to produce a pseudorandom number. The

-12-

pseudorandom number thus generated can be used by the GAN and the terminal as the secret key for the ensuing communications session.

The session key can be changed periodically to a different number in the pseudorandom number sequence. For example, the session key can be changed for a number of reasons, such as after a predetermined amount of data has been encrypted, or after traffic has been encrypted for a predetermined amount of time. The terminal or the GAN can initiate a change of the secret key, or the key can be changed according to a predetermined scheme or algorithm. For example, a request to change the secret session key can be implemented by transmitting a "session key change request" message, or by setting a "session key change request" bit in the header of a transmitted message.

Additionally, shorter session keys can be generated and less complicated encryption algorithms can be used with the pseudorandom number generation method described above. Consequently, a substantial amount of processing power can be saved in the GAN and especially in the terminal. The terminal can be configured to select the length of the session key to be used, in order to address trade offs between security and computational requirements. For example, the terminal's processor can select the length of a secret session key by generating a session key at that length, or by specifying the number of bits to be used from the output of the pseudorandom number generator. Alternatively, the terminal can specify the range of the output of the pseudorandom number generator to set a predetermined length.

Other alternative methods may be used to generate a pseudorandom number for a secret session key. For example, using a "Lagged Fibonacci" type of pseudorandom number generator, the n^{th} number in the pseudorandom number sequence, N_n , can be calculated as follows:

$$N_n = (N_{n-k} - N_{n-l}) \bmod M \quad (3)$$

where k and l are the so-called lags, and M defines the range of the pseudorandom numbers to be generated. For optimum results, the largest lag should be between 1000 and 10000. If a relatively long key is desired, a plurality of the pseudorandom numbers produced by equation 3 can be concatenated to produce a longer key. If

-13-

the pseudorandom numbers produced by equation 3 are to be floating point numbers between 0 and 1, M can be set to 1. The bit patterns of such floating point pseudorandom numbers can be used as symmetric encryption keys.

Another pseudorandom number generator that can be used to create a secret session key is based on an algorithm that produces pseudorandom numbers uniformly distributed between 0 and 1. Specifically, the seeds X_0 , Y_0 and Z_0 of the pseudorandom numbers N_n are initially set to integer values between 1 and 30000. The pseudorandom numbers N_n are then calculated as follows:

$$X_n = 171 * (X_{n-1} \bmod 177) - (2 * X_{n-1} / 177) \quad (4)$$

$$Y_n = 172 * (Y_{n-1} \bmod 176) - (35 * Y_{n-1} / 176) \quad (5)$$

$$Z_n = 170 * (Z_{n-1} \bmod 178) - (63 * Z_{n-1} / 178) \quad (6)$$

If any of the values of X_n , Y_n or Z_n are less than zero, respectively, then X_n is set equal to $X_n + 30269$, Y_n is set equal to $Y_n + 30307$, or Z_n is set equal to $Z_n + 30323$. The pseudorandom numbers N_n are then equal to $((X_n/30269 + Y_n/30307 + Z_n/30323) \bmod 1)$, where X_n , Y_n and Z_n are floating point numbers, and "amod" means that these numbers can be fractions. The floating point numbers generated with this algorithm form bit patterns that are suitable for use as symmetric encryption keys. The length of such keys can be extended by concatenating a plurality of the pseudorandom numbers generated.

Returning to the method illustrated by FIGURE 4, at step 208, preferably using the above-described RSA Algorithm, the terminal encrypts the secret symmetric key with the public key. For example, assume that the secret symmetric key generated at the terminal is represented by the letters SK. Using equation 1 of the RSA Algorithm, the secret key is encrypted as follows:

$$M^E \bmod PQ = > C$$

where (PQ, E) represents the public key, M is equal to SK, and C is the encrypted version of SK. The exponent E is set to 3.

In the preferred embodiment, the terminal places the encrypted secret key into a message format, which includes a header and message field. The header provides control information associated with the encrypted secret key that follows in the message field. A bit in the header can be set to indicate that the message field

-14-

that follows the header is encrypted. In other words, only the secret key field of the message is encrypted. The header of the message is transmitted in the clear. Consequently, a substantial amount of network processing time can be saved at the RNC, since the header indicates whether the subsequent message field is encrypted, and if so, only that portion of the message is to be decrypted.

5 At step 210, the terminal (118) transmits the encrypted secret key (C) to the GAN via the contacted base station (e.g., BS(1)). In the preferred embodiment, this secret key is used for the ensuing communications. Alternatively, at any time during the ensuing communications session, the terminal can generate a new secret key, encrypt it with the public key, and transmit the new encrypted secret key to the GAN. The security of the session is thereby increased, because by reducing the amount of time that a particular secret key is used for a session, the likelihood that the secret key will be broken by an unauthorized user is also reduced.

10 At step 212, the RNC (e.g., RNC(1)) receives the encrypted secret key (C) from the base station, and decrypts the secret key using the private key part of the RSA Algorithm. For example, using equation 2 (above) of the RSA Algorithm, the received encrypted secret key (C) is decrypted as follows:

$$C^D \text{ mod } PQ = > M$$

where (PQ, D) represents the private key, and M is equal to SK (secret key).

20 At step 214, the ensuing radio traffic between the RNC and the terminal is encrypted and decrypted with the secret key, which is now known to both the RNC and the terminal. A known symmetric encryption algorithm can be used to encrypt and decrypt the ensuing radio traffic with the secret key, such as, for example, a one, two or three pass Data Encryption Standard (DES) algorithm, or a Fast Encipherment Algorithm (FEAL).

25 As yet another encryption alternative, instead of using the RSA Algorithm to create a public/private key pair, a so-called Diffie-Hellman "exponential key exchange" algorithm can be used to let the terminal and the GAN agree on a secret session key. In using this encryption scheme, two numbers (α , q) are stored at the GAN. At the beginning of a communications session, the RNC transmits the two numbers directly (or broadcasts the numbers) to the terminal. The numbers α and

30

-15-

q are required to meet the following criteria: q is a large prime number that defines the finite (Galios) field $GF(q) = 1, 2, \dots, q-1$; and α is a fixed primitive element of $GF(q)$. In other words, the exponents (x) of $(\alpha^x \bmod q)$ produce all of the elements 1, 2, ..., q-1 of $GF(q)$. In order to generate an agreed to secret session key, the two numbers (α , q) are transmitted directly (or broadcast) from the GAN to the terminal. Alternatively, the two numbers can be already resident in the terminal's non-volatile memory. The terminal (118) generates the random number $X_T (1 < X_T < q-1)$, and computes the value of $Y_T = \alpha^{X_T} \bmod q$. The GAN (e.g., the RNC or base station) generates the random number $X_G (1 < X_G < q-1)$, and computes the value of $Y_G = \alpha^{X_G} \bmod q$. The random numbers can be generated at the terminal using the methods described above with respect to generating naturally occurring, truly random numbers.

Y_T and Y_G are transferred unencrypted to the respective GAN and terminal. Upon receipt of the number Y_G , the terminal calculates the value of $K_S = Y_G^{X_T} \bmod q = \alpha^{X_G X_T} \bmod q$. Upon receipt of the number Y_T , the GAN calculates the value of $K_S = Y_T^{X_G} \bmod q = \alpha^{X_T X_G} \bmod q$. The number X_T is kept secret at the terminal, the number X_G is kept secret at the GAN, but the value of K_S is now known at both the terminal and the GAN. The number K_S is therefore used by both as the communications session encryption key. An unauthorized user would not know either X_T or X_G and would have to compute the key K_S from Y_T and Y_G , which is a prohibitive computational process. A significant security advantage of using the exponential key exchange algorithm is that the GAN is not required to maintain secret private key data on a permanent basis.

In summary, when a communications session is first initiated between a GAN and a terminal, the terminal receives an asymmetric public key that has been continuously broadcast by the GAN, retrieved from the terminal's internal memory, or requested from the GAN. The GAN maintains a private key that can be used to decrypt information encrypted with the public key. The terminal generates and stores a naturally occurring random number as a secret session (symmetric) key, encrypts the symmetric session key with the public key, and transmits the encrypted session key to the GAN. The GAN decrypts the session key with the private key,

-16-

and both the GAN and terminal encrypt the ensuing communications with the secret session key. A primary technical advantage of transferring a public key from a GAN to a terminal at the onset of communications is that the GAN is not required to know the identity of the terminal in order to have encrypted communications with the terminal. However, a problem can arise if an unauthorized user attempts to impersonate a GAN and transmits a public key to the terminal. In that event, as described below, the terminal can be configured to authenticate the received public key and the identity of the GAN.

For example, when a public key is to be transferred from a GAN to a terminal, the key can be transferred with a public key "certificate". This certificate provides proof that the associated public key and the owner of that key are authentic. A "trusted" third party can issue the public key along with the certificate, which includes a "digital signature" that authenticates the third party's identity and the public key. The certificate can also contain the GAN's identity and the expiration date of the certificate, if any.

In one aspect of the invention, the GAN transmits the certificate and public key to the terminal. In that case, the public key of the third party is pre-stored (a priori) at the subscribing terminals.

FIGURE 5 is a block diagram of a method that can be used to certify the authenticity of a public key and the owner of the key with a digital signature, in accordance with the present invention. The method (300) of digitally signing a public key certificate and verifying its authenticity begins at step 302. At step 302, a "certificate" containing unencrypted information about the owner of the public key to be transferred to a terminal is prepared by a trusted third party. The unencrypted information also includes the public key and the expiration date of the certificate. At step 304, the resulting "unsigned" certificate is processed with an irreversible algorithm (e.g., a hashing algorithm) to produce a message digest at step 306, which is a digested or shortened version of the information included on the certificate. At step 308, the digest information is encrypted with a private key of a different public/private key pair. Preferably, an RSA algorithm similar to equations 1 and 2 above is used to derive this key pair. At step 310, a digitally signed public key

-17-

certificate is thereby produced that contains the originally unencrypted information (including the public key to be used for the communications session) and the digest information, which is now encrypted with the certificate issuer's private key. The digitally signed public key certificate is then transferred to the terminal that has initiated contact with the GAN.

At step 312, upon receiving the digitally signed certificate, the terminal's processor analyzes the unencrypted and encrypted portions of the document. At step 314, the unencrypted information is processed using an algorithm identical to the hashing algorithm used at step 304. At step 316, a second digested version of the unencrypted information is produced at the terminal. At step 318, the terminal's processor retrieves the pre-stored certificate issuer's public key from memory, and using an RSA algorithm, decrypts the encrypted digest information from the certificate. Another version of the unencrypted digested information is thereby produced at step 320. At step 322, the terminal compares the two versions of the unencrypted digested information, and if the compared information is identical, the certificate's signature and the session public key are assumed to be authentic. That certified public key can now be used by the terminal to encrypt the secret session key.

Although a preferred embodiment of the method and apparatus of the present invention has been illustrated in the accompanying Drawings and described in the foregoing Detailed Description, it will be understood that the invention is not limited to the embodiments disclosed, but is capable of numerous rearrangements, modifications and substitutions without departing from the spirit of the invention as set forth and defined by the following claims.

25

-18-

WHAT IS CLAIMED IS:

1. A method for encrypting communications traffic between a mobile communications network and a communications terminal, comprising the steps of:
storing a public key and a first identifier associated with said mobile
communications network at said communications terminal;
5 comparing said first identifier stored at said communications terminal with
a second identifier received from said mobile communications network and
producing a first predetermined result;
generating a secret key at said communications terminal;
10 encrypting said secret key with said stored public key at said communications
terminal; and
transmitting said encrypted secret key from said communications terminal.
2. The method according to Claim 1, further comprising the steps of:
15 receiving said encrypted secret key at said mobile communications network;
decrypting said received encrypted secret key with a private key, said private
key associated with said public key; and
encrypting said communications traffic with said secret key.
- 20 3. The method according to Claim 1, wherein the step of storing a
public key comprises the step of a priori pre-storing the public key.
4. The method according to Claim 1, further comprising the step of
transmitting said public key from said mobile communications network upon
25 receiving a public key request from said communications terminal.
5. The method according to Claim 4, wherein the step of transmitting
said public key further comprises the step of transmitting information to authenticate
said public key.

30

-19-

6. The method according to Claim 4, further comprising the step of transmitting said request from said communications terminal upon said comparing step producing a second predetermined result.

5 7. The method according to Claim 1, wherein the steps of receiving and decrypting said encrypted secret key are performed at a radio base station in said mobile communications network.

8. The method according to Claim 1, wherein the step of decrypting said received encrypted secret key is performed at a radio network controller in said mobile communications network.

9. The method according to Claim 1, wherein said mobile communications network comprises a generic communications network.

15 10. The method according to Claim 1, wherein said communications terminal comprises a mobile terminal.

11. The method according to Claim 1, wherein said communications terminal comprises a fixed terminal.

12. The method according to Claim 1, wherein said communications terminal comprises an unidentified communications terminal.

25 13. The method according to Claim 1, wherein said mobile communications network comprises a cellular phone network.

14. The method according to Claim 1, further comprising the steps of: connecting a plurality of service networks to said mobile communications network, a user of said communications terminal being a subscriber to at least one of said plurality of service networks; and

-20-

providing a communications path between said communications terminal and said at least one of said plurality of service networks.

5 15. The method according to Claim 1, wherein said private key and said public key are associated by an RSA Algorithm.

16. The method according to Claim 1, wherein said secret key comprises a symmetric encryption key.

10 17. The method according to Claim 1, wherein the step of generating a secret key comprises the step of generating a naturally occurring random number.

15 18. The method according to Claim 1, wherein the step of generating a secret key comprises the steps of:
detecting a received signal in digital form at said communications terminal;
and
extracting at least one low order bit from said detected received signal.

20 19. The method according to Claim 1, wherein the step of generating a secret key comprises the steps of:
detecting a signal at an output of a microphone A/D converter; and
extracting at least one low order bit from said detected output signal.

25 20. The method according to Claim 1, wherein the step of generating a secret key comprises the steps of:
detecting a signal at an output of a speech codec; and
extracting at least one low order bit from said detected output signal.

30 21. The method according to Claim 1, wherein the step of generating a secret key comprises the steps of:

-21-

generating a seed for a pseudorandom number; and
generating a pseudorandom number from said seed.

22. The method according to Claim 1, wherein a length of said secret key
5 is predetermined at said communications terminal.

23. The method according to Claim 1, wherein said secret key further
comprises a plurality of concatenated numbers.

10 24. The method according to Claim 1, wherein the step of storing said
public key and said first identifier further comprises storing an expiration date
associated with said public key.

15 25. The method according to Claim 24, wherein said communications
terminal transmits a public key request to said mobile communications network if
said public key has expired.

20 26. The method according to Claim 1, further comprising the steps of:
changing said public key at said mobile communications network; and
storing said changed public key at said communications terminal.

25 27. The method according to Claim 26, wherein the step of changing said
public key further comprises the step of broadcasting said changed public key from
said mobile communications network for a predetermined period of time.

28. A method for encrypting traffic between a generic
communications network and a first communications terminal, comprising the steps
of:

30 broadcasting a public key from said generic communications network to a
plurality of communications terminals, said plurality of communications terminals
including said first communications terminal;

-22-

generating a secret key at said first communications terminal;
encrypting said secret key with said public key at said first communications terminal;
transmitting said encrypted secret key from said first communications terminal;
receiving said encrypted secret key at said generic communications network;
decrypting said received encrypted secret key with a private key, said private key associated with said public key; and
encrypting said traffic with said secret key.

29. The method according to Claim 28, wherein the broadcasting step further comprises the steps of:

transferring said public key from a radio network controller to at least one base station in said generic communications network; and

transmitting said public key from said at least one base station.

30. The method according to Claim 28, wherein said broadcasting step comprises the step of transmitting said public key from a plurality of base stations in said generic communications network.

31. The method according to Claim 28, wherein said first communications terminal comprises an unidentified communications terminal.

32. The method according to Claim 28, wherein the step of broadcasting said public key further comprises the step of broadcasting information to authenticate said public key.

33. The method according to Claim 28, wherein the step of broadcasting said public key further comprises the step of transmitting, on request, information to authenticate said public key.

-23-

34. A method for encrypting communications traffic between a mobile communications network and a communications terminal, comprising the steps of:

storing two numbers associated with a Diffie-Hellman exponential key exchange algorithm and a first identifier associated with said mobile communications network at said communications terminal;

5 comparing said first identifier stored at said communications terminal with a second identifier received from said mobile communications network and producing a first predetermined result;

generating a first random number at said communications terminal;

10 generating a second random number at said mobile communications network;

and

using said first and second random numbers as inputs to said Diffie-Hellman exponential key exchange algorithm, generating a third number to be used as a secret key by said communications terminal and said mobile communications network.

15

35. The method according to Claim 34, wherein the step of storing two numbers comprises the step of a priori pre-storing said two numbers.

20 36. The method according to Claim 34, further comprising the step of transmitting said two numbers from said mobile communications network upon receiving a request for said two numbers from said communications terminal.

25 37. The method according to Claim 36, further comprising the step of transmitting said request from said communications terminal upon said comparing step producing a second predetermined result.

30 38. The method according to Claim 34, wherein the step of storing said two numbers and said first identifier further comprises storing an expiration date associated with said two numbers.

-24-

39. The method according to Claim 38, wherein said communications terminal transmits a request for two new numbers associated with said Diffie-Hellman exponential key exchange algorithm if said two numbers has expired.

5 40. The method according to Claim 34, further comprising the steps of:
changing said two numbers associated with a Diffie-Hellman exponential key
exchange algorithm at said mobile communications network; and
storing said changed two numbers at said communications terminal.

10 41. The method according to Claim 40, wherein the step of changing said
two numbers further comprises the step of broadcasting said changed two numbers
from said mobile communications network for a predetermined period of time.

15 42. A method for encrypting traffic between a generic communications
network and a first communications terminal, comprising the steps of:

broadcasting two numbers associated with an exponential key exchange
algorithm from said generic communications network to a plurality of
communications terminals, said plurality of communications terminals including said
first communications terminal;

20 generating a first random number at said first communications terminal;
generating a second random number at said generic communications network;
using said first and second random numbers as inputs to said exponential key
exchange algorithm, generating a third number to be used as a secret key by said
first communications terminal and said generic communications network;
25 and encrypting said traffic with said secret key.

43. A system for use in encrypting traffic between a generic
communications network and a communications terminal, comprising:

30 an access network included in said generic communications network; and
access network means coupled to said communications terminal and
associated with said access network, for storing a public encryption key associated

-25-

with said generic communications network, generating a secret key, encrypting said secret key with said stored public encryption key, and transmitting said encrypted secret key to said generic communications network.

5 44. A system for use in encrypting traffic between a generic communications network and a communications terminal, comprising:

first network means for storing a private encryption key, distributing a public encryption key, and decrypting an encrypted secret session key;

10 second network means connected to said first network means, for broadcasting said distributed public encryption key, said first and second network means associated with an access network of said generic communications network; and

15 access network means coupled to said communications terminal and associated with said access network of said generic communications network, for receiving said broadcast public encryption key, generating a secret key, encrypting said secret key with said received public encryption key, and transmitting said encrypted secret key to said generic communications network.

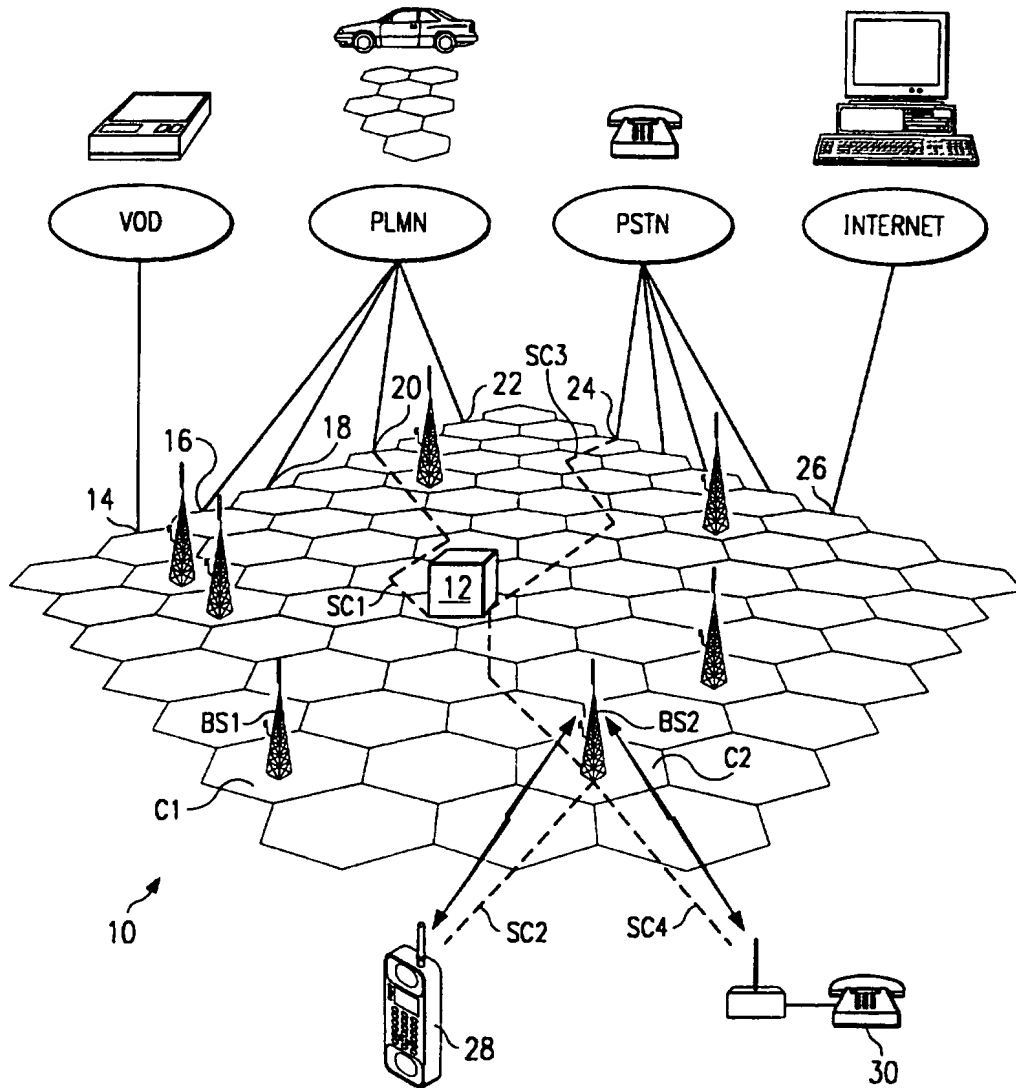


FIG. 1

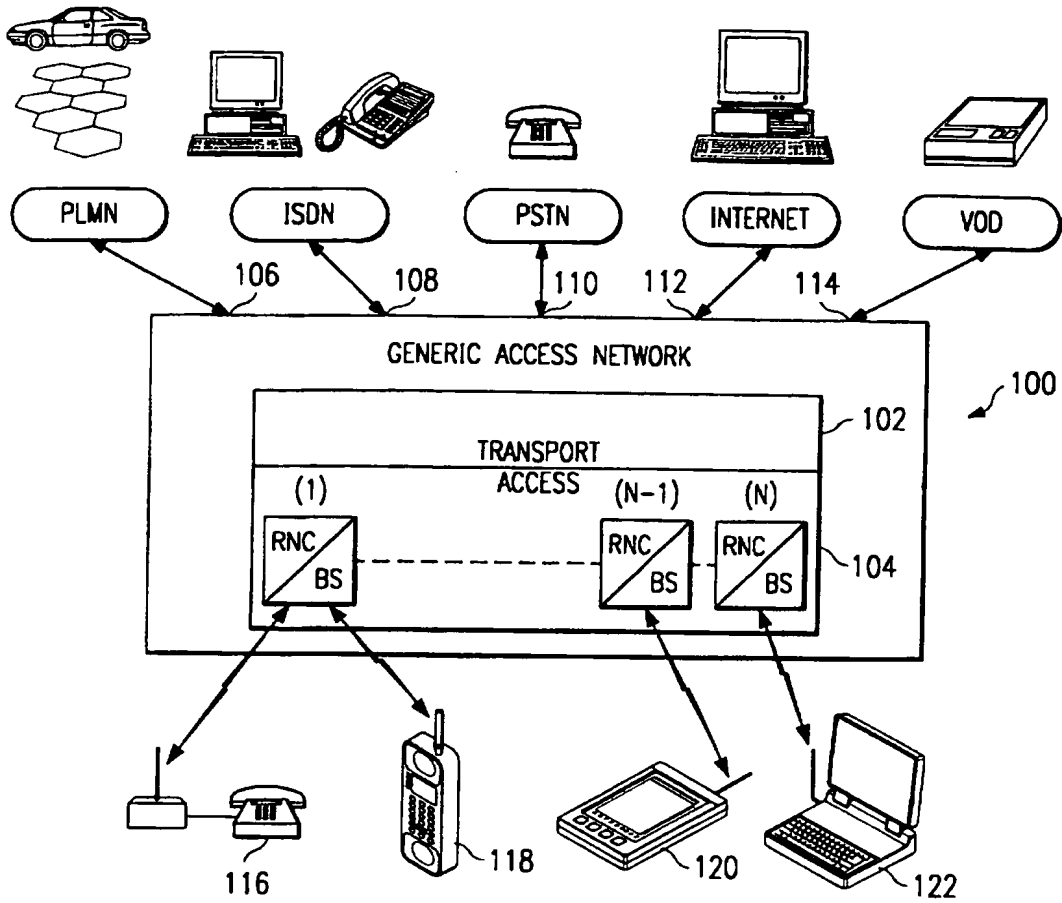


FIG. 2

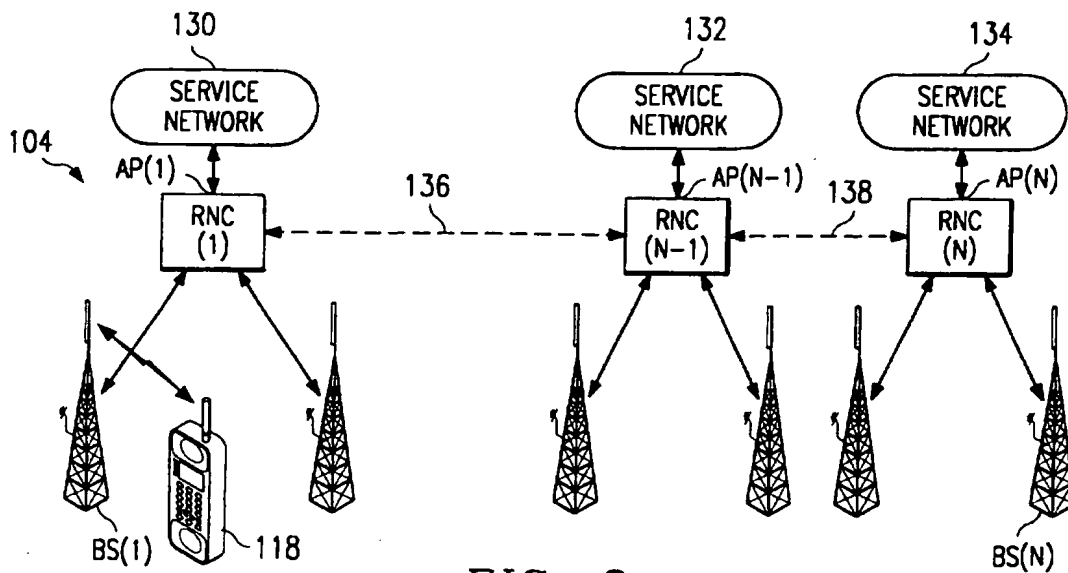
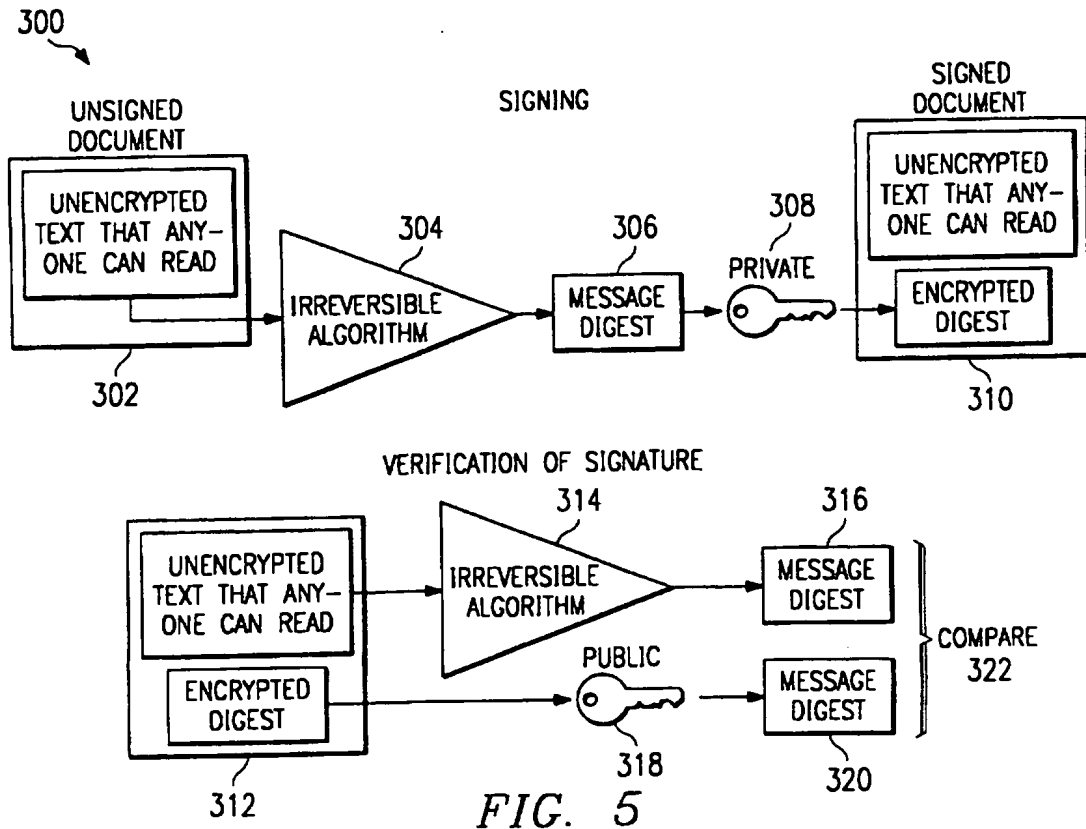
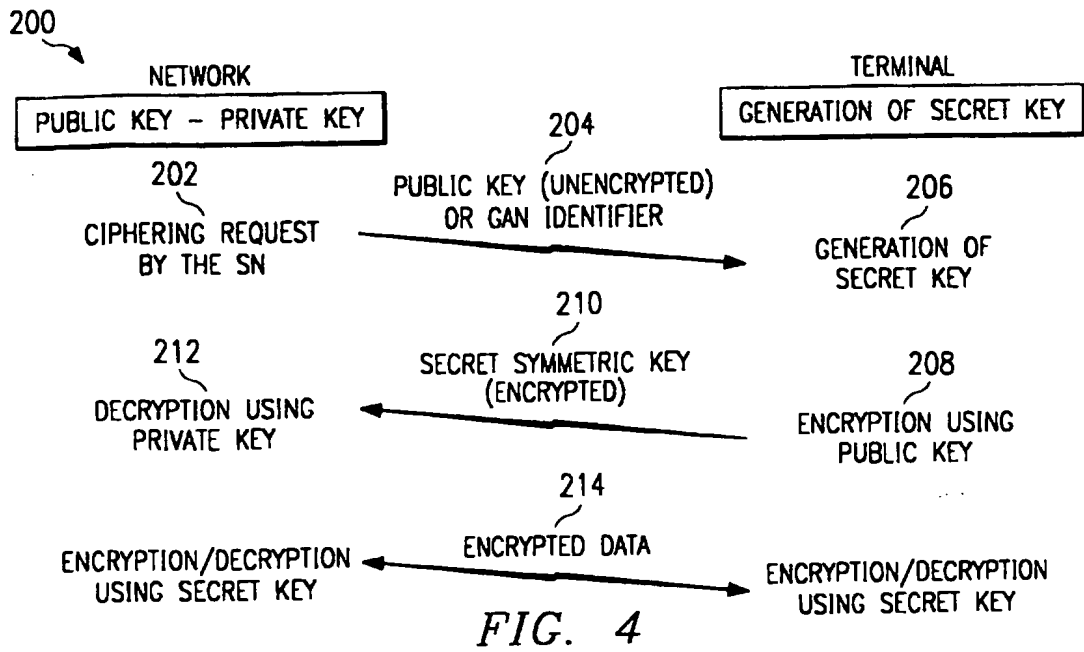


FIG. 3



INTERNATIONAL SEARCH REPORT

International Application No

PCT/SE 97/01407

A. CLASSIFICATION OF SUBJECT MATTER
 IPC 6 H04L9/08 H0407/32

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
 IPC 6 H04L H04Q

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 222 140 A (BELLER ET AL.) 22 June 1993 see column 4, line 57 - column 5, line 3 see column 5, line 13 - line 37 ---	1, 2, 7, 10, 28, 29, 34, 42-44
A	GB 2 297 016 A (KOKUSAI DENSHIN DENWA) 17 July 1996 see page 19, line 11 - page 21, line 2; figure 7 --- -/--	28, 44

Further documents are listed in the continuation of box C.

Patent family members are listed in annex.

* Special categories of cited documents :

- "A" document defining the general state of the art which is not considered to be of particular relevance
- "E" earlier document but published on or after the international filing date
- "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- "O" document referring to an oral disclosure, use, exhibition or other means
- "P" document published prior to the international filing date but later than the priority date claimed

- "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- "&" document member of the same patent family

Date of the actual completion of the international search

19 November 1997

Date of mailing of the international search report

02/12/1997

Name and mailing address of the ISA
 European Patent Office, P.B. 5818 Patentaan 2
 NL - 2280 HV Rijswijk
 Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,
 Fax: (+31-70) 340-3016

Authorized officer

Holper, G

INTERNATIONAL SEARCH REPORT

International Application No

PCT/SE 97/01407

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	<p>MEVEL F ET AL: "Distributed communication services in the Masix system" CONFERENCE PROCEEDINGS OF THE 1996 IEEE FIFTEENTH ANNUAL INTERNATIONAL PHOENIX CONFERENCE ON COMPUTERS AND COMMUNICATIONS (CAT. NO.96CH35917), CONFERENCE PROCEEDINGS OF THE 1996 IEEE FIFTEENTH ANNUAL INTERNATIONAL PHOENIX CONFERENCE ON COMPUTERS AND, ISBN 0-7803-3255-5, 1996, NEW YORK, NY, USA, IEEE, USA, pages 172-178, XP000594787 see page 174, right-hand column, line 25 - line 29 see page 176, right-hand column, line 26 - page 177, left-hand column, last line; figure 5</p>	28, 43, 44
A	<p style="text-align: center;">---</p> <p>PATENT ABSTRACTS OF JAPAN vol. 95, no. 008 & JP 07 203540 A (N T T IDOU TSUUSHINMOU KK), 4 August 1995, see abstract</p>	1, 34
A	<p style="text-align: center;">---</p> <p>EP 0 067 977 A (SIEMENS) 29 December 1982 see abstract; figure 2</p> <p style="text-align: center;">-----</p>	24

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

PCT/SE 97/01407

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5222140 A	22-06-93	NONE	
GB 2297016 A	17-07-96	JP 8195741 A	30-07-96
EP 67977 A	29-12-82	DE 3123167 C	24-02-83



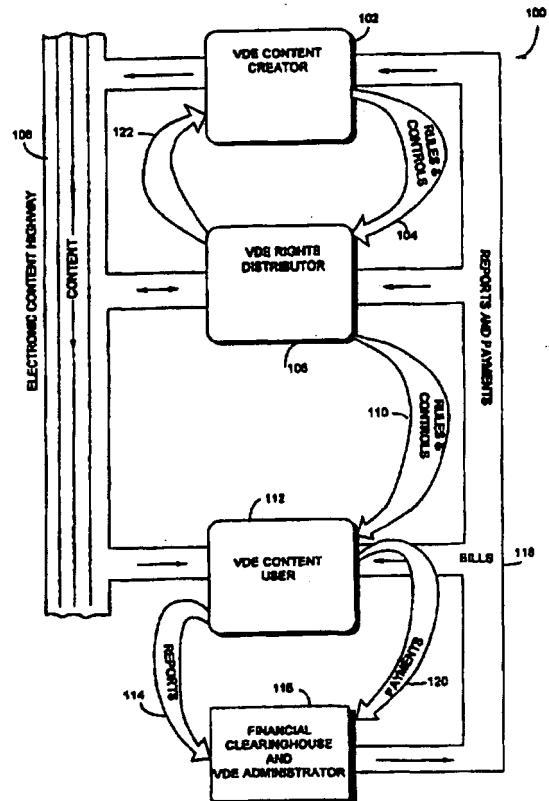
INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<p>(51) International Patent Classification ⁶ : G06F 1/00</p>	<p>A1</p>	<p>(11) International Publication Number: WO 98/09209 (43) International Publication Date: 5 March 1998 (05.03.98)</p>
<p>(21) International Application Number: PCT/US97/15243 (22) International Filing Date: 29 August 1997 (29.08.97) (30) Priority Data: 08/706,206 30 August 1996 (30.08.96) US (71) Applicant: INTERTRUST TECHNOLOGIES CORP. [US/US]; 460 Oakmead Parkway, Sunnyvale, CA 94086 (US). (72) Inventors: GINTER, Karl, L.; 10404 43rd Avenue, Beltsville, MD 20705 (US). SHEAR, Victor, H.; 5203 Battery Lane, Bethesda, MD 20814 (US). SIBERT, W., Olin; 30 Ingleside Road, Lexington, MA 02173-2522 (US). SPAHN, Francis, J.; 2410 Edwards Avenue, El Cerrito, CA 94530 (US). VAN WIE, David, M.; 1250 Lakeside Drive, Sunnyvale, CA 94086 (US). (74) Agent: FARIS, Robert, W.; Nixon & Vanderhye P.C., 8th floor, 1100 North Glebe Road, Arlington, VA 22201-4714 (US).</p>	<p>(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).</p> <p>Published <i>With international search report.</i> <i>Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</i></p>	

(54) Title: **SYSTEMS AND METHODS FOR SECURE TRANSACTION MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION**

(57) Abstract

The present invention provides systems and methods for electronic commerce including secure transaction management and electronic rights protection. Electronic appliances such as computers employed in accordance with the present invention help to ensure that information is accessed and used only in authorized ways, and maintain the integrity, availability, and/or confidentiality of the information. Secure subsystems used with such electronic appliances provide a distributed virtual distribution environment (VDE) that may enforce a secure chain of handling and control, for example, to control and/or meter or otherwise monitor use of electronically stored or disseminated information. Such a virtual distribution environment may be used to protect rights of various participants in electronic commerce and other electronic or electronic-facilitated transactions. Secure distributed and other operating system environments and architectures, employing, for example, secure semiconductor processing arrangements that may establish secure, protected environments at each node. These techniques may be used to support an end-to-end electronic information distribution capability that may be used, for example, utilizing the "electronic highway".



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakistan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LJ	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

**SYSTEMS AND METHODS FOR SECURE TRANSACTION
MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION**

Field(s) of the Invention(s)

This invention generally relates to computer and/or
electronic security.

5

More particularly, this invention relates to systems and
techniques for secure transaction management. This invention
also relates to computer-based and other electronic appliance-
based technologies that help to ensure that information is
10 accessed and/or otherwise used only in authorized ways, and
maintains the integrity, availability, and/or confidentiality of
such information and processes related to such use.

10

The invention also relates to systems and methods for
15 protecting rights of various participants in electronic commerce
and other electronic or electronically-facilitated transactions.

15

The invention also relates to secure chains of handling and
control for both information content and information employed to
20 regulate the use of such content and consequences of such use. It
also relates to systems and techniques that manage, including
meter and/or limit and/or otherwise monitor use of electronically
stored and/or disseminated information. The invention

20

particularly relates to transactions, conduct and arrangements that make use of, including consequences of use of, such systems and/or techniques.

5 The invention also relates to distributed and other operating systems, environments and architectures. It also generally relates to secure architectures, including, for example, tamper-resistant hardware-based processors, that can be used to establish security at each node of a distributed system.

10

Background and Summary of the Invention(s)

Telecommunications, financial transactions, government processes, business operations, entertainment, and personal business productivity all now depend on electronic appliances. Millions of these electronic appliances have been electronically connected together. These interconnected electronic appliances comprise what is increasingly called the "information highway." Many businesses, academicians, and government leaders are concerned about how to protect the rights of citizens and organizations who use this information (also "electronic" or

15

20

"digital") highway.

Electronic Content

Today, virtually anything that can be represented by words, numbers, graphics, or system of commands and instructions can be formatted into electronic digital information.

5 Television, cable, satellite transmissions, and on-line services transmitted over telephone lines, compete to distribute digital information and entertainment to homes and businesses. The owners and marketers of this content include software developers, motion picture and recording companies, publishers
10 of books, magazines, and newspapers, and information database providers. The popularization of on-line services has also enabled the individual personal computer user to participate as a content provider. It is estimated that the worldwide market for electronic information in 1992 was approximately \$40 billion and
15 is expected to grow to \$200 billion by 1997, according to Microsoft Corporation. The present invention can materially enhance the revenue of content providers, lower the distribution costs and the costs for content, better support advertising and usage information gathering, and better satisfy the needs of
20 electronic information users. These improvements can lead to a significant increase in the amount and variety of electronic information and the methods by which such information is distributed.

The inability of conventional products to be shaped to the needs of electronic information providers and users is sharply in contrast to the present invention. Despite the attention devoted by a cross-section of America's largest telecommunications, computer, entertainment and information provider companies to some of the problems addressed by the present invention, only the present invention provides commercially secure, effective solutions for configurable, general purpose electronic commerce transaction/distribution control systems.

10

Controlling Electronic Content

The present invention provides a new kind of "virtual distribution environment" (called "VDE" in this document) that secures, administers, and audits electronic information use. VDE also features fundamentally important capabilities for managing content that travels "across" the "information highway." These capabilities comprise a rights protection solution that serves all electronic community members. These members include content creators and distributors, financial service providers, end-users, and others. VDE is the first general purpose, configurable, transaction control/rights protection solution for users of computers, other electronic appliances, networks, and the information highway.

15

20

A fundamental problem for electronic content providers is extending their ability to control the use of proprietary information. Content providers often need to limit use to authorized activities and amounts. Participants in a business model involving, for example, provision of movies and advertising on optical discs may include actors, directors, script and other writers, musicians, studios, publishers, distributors, retailers, advertisers, credit card services, and content end-users. These participants need the ability to embody their range of agreements and requirements, including use limitations, into an “extended” agreement comprising an overall electronic business model. This extended agreement is represented by electronic content control information that can automatically enforce agreed upon rights and obligations. Under VDE, such an extended agreement may comprise an electronic contract involving all business model participants. Such an agreement may alternatively, or in addition, be made up of electronic agreements between subsets of the business model participants. Through the use of VDE, electronic commerce can function in the same way as traditional commerce—that is commercial relationships regarding products and services can be shaped through the negotiation of one or more agreements between a variety of parties.

Commercial content providers are concerned with ensuring proper compensation for the use of their electronic information. Electronic digital information, for example a CD recording, can today be copied relatively easily and inexpensively. Similarly, 5 unauthorized copying and use of software programs deprives rightful owners of billions of dollars in annual revenue according to the International Intellectual Property Alliance. Content providers and distributors have devised a number of limited function rights protection mechanisms to protect their rights. 10 Authorization passwords and protocols, license servers, "lock/unlock" distribution methods, and non-electronic contractual limitations imposed on users of shrink-wrapped software are a few of the more prevalent content protection schemes. In a commercial context, these efforts are inefficient 15 and limited solutions.

Providers of "electronic currency" have also created protections for their type of content. These systems are not sufficiently adaptable, efficient, nor flexible enough to support 20 the generalized use of electronic currency. Furthermore, they do not provide sophisticated auditing and control configuration capabilities. This means that current electronic currency tools lack the sophistication needed for many real-world financial business models. VDE provides means for anonymous currency

and for "conditionally" anonymous currency, wherein currency related activities remain anonymous except under special circumstances.

5

VDE Control Capabilities

VDE allows the owners and distributors of electronic digital information to reliably bill for, and securely control, audit, and budget the use of, electronic information. It can reliably detect and monitor the use of commercial information products. VDE uses a wide variety of different electronic information delivery means: including, for example, digital networks, digital broadcast, and physical storage media such as optical and magnetic disks. VDE can be used by major network providers, hardware manufacturers, owners of electronic information, providers of such information, and clearinghouses that gather usage information regarding, and bill for the use of, electronic information.

20

VDE provides comprehensive and configurable transaction management, metering and monitoring technology. It can change how electronic information products are protected, marketed, packaged, and distributed. When used, VDE should result in higher revenues for information providers and greater

user satisfaction and value. Use of VDE will normally result in lower usage costs, decreased transaction costs, more efficient access to electronic information, re-usability of rights protection and other transaction management implementations, greatly
5 improved flexibility in the use of secured information, and greater standardization of tools and processes for electronic transaction management. VDE can be used to create an adaptable environment that fulfills the needs of electronic information owners, distributors, and users; financial
10 clearinghouses; and usage information analyzers and resellers.

Rights and Control Information

In general, the present invention can be used to protect the rights of parties who have:

15

(a) proprietary or confidentiality interests in electronic information. It can, for example, help ensure that information is used only in authorized ways;

20

(b) financial interests resulting from the use of electronically distributed information. It can help ensure that content providers will be paid for use of distributed information; and

- (c) interests in electronic credit and electronic currency storage, communication, and/or use including electronic cash, banking, and purchasing.

5 Protecting the rights of electronic community members involves a broad range of technologies. VDE combines these technologies in a way that creates a "distributed" electronic rights protection "environment." This environment secures and protects transactions and other processes important for rights
10 protection. VDE, for example, provides the ability to prevent, or impede, interference with and/or observation of, important rights related transactions and processes. VDE, in its preferred embodiment, uses special purpose tamper resistant Secure Processing Units (SPUs) to help provide a high level of security
15 for VDE processes and information storage and communication.

The rights protection problems solved by the present invention are electronic versions of basic societal issues. These issues include protecting property rights, protecting privacy
20 rights, properly compensating people and organizations for their work and risk, protecting money and credit, and generally protecting the security of information. VDE employs a system that uses a common set of processes to manage rights issues in an efficient, trusted, and cost-effective way.

VDE can be used to protect the rights of parties who create electronic content such as, for example: records, games, movies, newspapers, electronic books and reference materials, personal electronic mail, and confidential records and communications.

5 The invention can also be used to protect the rights of parties who provide electronic products, such as publishers and distributors; the rights of parties who provide electronic credit and currency to pay for use of products, for example, credit clearinghouses and banks; the rights to privacy of parties who
10 use electronic content (such as consumers, business people, governments); and the privacy rights of parties *described* by electronic information, such as privacy rights related to information contained in a medical record, tax record, or personnel record.

15

In general, the present invention can protect the rights of parties who have:

(a) commercial interests in electronically distributed information -- the present invention can help
20 ensure, for example, that parties, will be paid for use of distributed information in a manner consistent with their agreement;

- (b) proprietary and/or confidentiality interests in electronic information -- the present invention can, for example, help ensure that data is used only in authorized ways;

5

- (c) interests in electronic credit and electronic currency storage, communication, and/or use -- this can include electronic cash, banking, and purchasing; and

10

- (d) interests in electronic information derived, at least in part, from use of other electronic information.

VDE Functional Properties

15

VDE is a cost-effective and efficient rights protection solution that provides a unified, consistent system for securing and managing transaction processing. VDE can:

- (a) audit and analyze the use of content,
- (b) ensure that content is used only in authorized ways, and

20

- (c) allow information regarding content usage to be used only in ways approved by content users.

In addition, VDE:

5

- (a) is very configurable, modifiable, and re-usable;
- (b) supports a wide range of useful capabilities that may be combined in different ways to accommodate most potential applications;

10

- (c) operates on a wide variety of electronic appliances ranging from hand-held inexpensive devices to large mainframe computers;

15

- (d) is able to ensure the various rights of a number of different parties, and a number of different rights protection schemes, simultaneously;

20

- (e) is able to preserve the rights of parties through a series of transactions that may occur at different times and different locations;

(f) is able to flexibly accommodate different ways of securely delivering information and reporting usage; and

5 (g) provides for electronic analogues to "real" money and credit, including anonymous electronic cash, to pay for products and services and to support personal (including home) banking and other financial activities.

10

VDE economically and efficiently fulfills the rights protection needs of electronic community members. Users of VDE will not require additional rights protection systems for different information highway products and rights
15 problems—nor will they be required to install and learn a new system for each new information highway application.

20

VDE provides a unified solution that allows all content creators, providers, and users to employ the same electronic rights protection solution. Under authorized circumstances, the participants can freely exchange content and associated content control sets. This means that a user of VDE may, if allowed, use the same electronic system to work with different kinds of content having different sets of content control information. The

content and control information supplied by one group can be
used by people who normally use content and control information
supplied by a different group. VDE can allow content to be
exchanged "universally" and users of an implementation of the
5 present invention can interact electronically without fear of
incompatibilities in content control, violation of rights, or the
need to get, install, or learn a new content control system.

The VDE securely administers transactions that specify
10 protection of rights. It can protect electronic rights including, for
example:

- (a) the property rights of authors of electronic content,
- 15 (b) the commercial rights of distributors of content,
- (c) the rights of any parties who facilitated the
distribution of content,
- 20 (d) the privacy rights of users of content,
- (e) the privacy rights of parties portrayed by stored
and/or distributed content, and

- (f) any other rights regarding enforcement of electronic agreements.

5 VDE can enable a very broad variety of electronically enforced commercial and societal agreements. These agreements can include electronically implemented contracts, licenses, laws, regulations, and tax collection.

Contrast With Traditional Solutions

10 Traditional content control mechanisms often require users to purchase more electronic information than the user needs or desires. For example, infrequent users of shrink-wrapped software are required to purchase a program at the same price as frequent users, even though they may receive
15 much less value from their less frequent use. Traditional systems do not scale cost according to the extent or character of usage and traditional systems can not attract potential customers who find that a fixed price is too high. Systems using
20 traditional mechanisms are also not normally particularly secure. For example, shrink-wrapping does not prevent the constant illegal pirating of software once removed from either its physical or electronic package.

Traditional electronic information rights protection systems are often inflexible and inefficient and may cause a content provider to choose costly distribution channels that increase a product's price. In general these mechanisms restrict product pricing, configuration, and marketing flexibility. These compromises are the result of techniques for controlling information which cannot accommodate both different content models and content models which reflect the many, varied requirements, such as content delivery strategies, of the model participants. This can limit a provider's ability to deliver sufficient overall value to justify a given product's cost in the eyes of many potential users. VDE allows content providers and distributors to create applications and distribution networks that reflect content providers' and users' preferred business models. It offers users a uniquely cost effective and feature rich system that supports the ways providers *want* to distribute information and the ways users *want* to use such information. VDE supports content control models that ensure rights and allow content delivery strategies to be shaped for maximum commercial results.

Chain of Handling and Control

VDE can protect a collection of rights belonging to various parties having in rights in, or to, electronic information. This

information may be at one location or dispersed across (and/or moving between) multiple locations. The information may pass through a "chain" of distributors and a "chain" of users. Usage information may also be reported through one or more "chains" of parties. In general, VDE enables parties that (a) have rights in electronic information, and/or (b) act as direct or indirect agents for parties who have rights in electronic information, to ensure that the moving, accessing, modifying, or otherwise using of information can be securely controlled by rules regarding how, when, where, and by whom such activities can be performed.

VDE Applications and Software

VDE is a secure system for regulating electronic conduct and commerce. Regulation is ensured by control information put in place by one or more parties. These parties may include content providers, electronic hardware manufacturers, financial service providers, or electronic "infrastructure" companies such as cable or telecommunications companies. The control information implements "Rights Applications." Rights applications "run on" the "base software" of the preferred embodiment. This base software serves as a secure, flexible, general purpose foundation that can accommodate many different rights applications, that is, many different business models and their respective participant requirements.

A rights application under VDE is made up of special purpose pieces, each of which can correspond to one or more basic electronic processes needed for a rights protection environment. These processes can be combined together like building blocks to create electronic agreements that can protect the rights, and may enforce fulfillment of the obligations, of electronic information users and providers. One or more providers of electronic information can easily combine selected building blocks to create a rights application that is unique to a specific content distribution model. A group of these pieces can represent the capabilities needed to fulfill the agreement(s) between users and providers. These pieces accommodate many requirements of electronic commerce including:

- the distribution of permissions to use electronic information;
- the persistence of the control information and sets of control information managing these permissions;
- configurable control set information that can be selected by users for use with such information;

- data security and usage auditing of electronic information; and
- a secure system for currency, compensation and debit management.

5

10

15

For electronic commerce, a rights application, under the preferred embodiment of the present invention, can provide electronic enforcement of the business agreements between all participants. Since different groups of components can be put together for different applications, the present invention can provide electronic control information for a wide variety of different products and markets. This means the present invention can provide a "unified," efficient, secure, and cost-effective system for electronic commerce and data security. This allows VDE to serve as a single standard for electronic rights protection, data security, and electronic currency and banking.

20

In a VDE, the separation between a rights application and its foundation permits the efficient selection of sets of control information that are appropriate for each of many different types of applications and uses. These control sets can reflect both rights of electronic community members, as well as obligations

(such as providing a history of one's use of a product or paying taxes on one's electronic purchases). VDE flexibility allows its users to electronically implement and enforce common social and commercial ethics and practices. By providing a unified control system, the present invention supports a vast range of possible transaction related interests and concerns of individuals, communities, businesses, and governments. Due to its open design, VDE allows (normally under securely controlled circumstances) applications using technology independently created by users to be "added" to the system and used in conjunction with the foundation of the invention. In sum, VDE provides a system that can fairly reflect and enforce agreements among parties. It is a broad ranging and systematic solution that answers the pressing need for a secure, cost-effective, and fair electronic environment.

VDE Implementation

The preferred embodiment of the present invention includes various tools that enable system designers to directly insert VDE capabilities into their products. These tools include an Application Programmer's Interface ("API") and a Rights Permissioning and Management Language ("RPML"). The RPML provides comprehensive and detailed control over the use of the invention's features. VDE also includes certain user

interface subsystems for satisfying the needs of content providers, distributors, and users.

Information distributed using VDE may take many forms.

5 It may, for example, be "distributed" for use on an individual's own computer, that is the present invention can be used to provide security for locally stored data. Alternatively, VDE may be used with information that is dispersed by authors and/or publishers to one or more recipients. This information may take
10 many forms including: movies, audio recordings, games, electronic catalog shopping, multimedia, training materials, E-mail and personal documents, object oriented libraries, software programming resources, and reference/record keeping information resources (such as business, medical, legal,
15 scientific, governmental, and consumer databases).

Electronic rights protection provided by the present invention will also provide an important foundation for trusted and efficient home and commercial banking, electronic credit
20 processes, electronic purchasing, true or conditionally anonymous electronic cash, and EDI (Electronic Data Interchange). VDE provides important enhancements for improving data security in organizations by providing "smart"

transaction management features that can be far more effective than key and password based "go/no go" technology.

5 VDE normally employs an integration of cryptographic and other security technologies (e.g. encryption, digital signatures, etc.), with other technologies including: component, distributed, and event driven operating system technology, and related communications, object container, database, smart agent, smart card, and semiconductor design technologies.

10

I. Overview

A. VDE Solves Important Problems and Fills Critical Needs

15 The world is moving towards an integration of electronic information appliances. This interconnection of appliances provides a foundation for much greater electronic interaction and the evolution of electronic commerce. A variety of capabilities are required to implement an electronic commerce environment. VDE is the first system that provides many of these capabilities and therefore solves fundamental problems related to electronic
20 dissemination of information.

Electronic Content

VDE allows electronic arrangements to be created involving two or more parties. These agreements can themselves comprise a collection of agreements between participants in a commercial value chain and/or a data security chain model for handling, auditing, reporting, and payment. It can provide efficient, reusable, modifiable, and consistent means for secure electronic content: distribution, usage control, usage payment, usage auditing, and usage reporting. Content may, for example, include:

- financial information such as electronic currency and credit;
- commercially distributed electronic information such as reference databases, movies, games, and advertising; and
- electronic properties produced by persons and organizations, such as documents, e-mail, and proprietary database information.

VDE enables an electronic commerce marketplace that supports differing, competitive business partnerships, agreements, and evolving overall business models.

5 The features of VDE allow it to function as the first
trusted electronic information control environment that can
conform to, and support, the bulk of conventional electronic
commerce and data security requirements. In particular, VDE
enables the participants in a business value chain model to
10 create an electronic version of traditional business agreement
terms and conditions and further enables these participants to
shape and evolve their electronic commerce models as they
believe appropriate to their business requirements.

15 VDE offers an architecture that avoids reflecting specific
distribution biases, administrative and control perspectives, and
content types. Instead, VDE provides a broad-spectrum,
fundamentally configurable and portable, electronic transaction
control, distributing, usage, auditing, reporting, and payment
20 operating environment. VDE is not limited to being an
application or application specific toolset that covers only a
limited subset of electronic interaction activities and
participants. Rather, VDE supports systems by which such
applications can be created, modified, and/or reused. As a result,

the present invention answers pressing, unsolved needs by offering a system that supports a standardized control environment which facilitates interoperability of electronic appliances, interoperability of content containers, and efficient
5 creation of electronic commerce applications and models through the use of a programmable, secure electronic transactions management foundation and reusable and extensible executable components. VDE can support a single electronic "world" within which most forms of electronic transaction activities can be
10 managed.

To answer the developing needs of rights owners and content providers and to provide a system that can accommodate the requirements and agreements of all parties that may be
15 involved in electronic business models (creators, distributors, administrators, users, credit providers, etc.), VDE supplies an efficient, largely transparent, low cost and sufficiently secure system (supporting both hardware/ software and software only models). VDE provides the widely varying secure control and
20 administration capabilities required for:

1. Different types of electronic content,
2. Differing electronic content delivery schemes,

3. Differing electronic content usage schemes,
4. Different content usage platforms, and
5. Differing content marketing and model strategies.

VDE may be combined with, or integrated into, many separate computers and/or other electronic appliances. These appliances typically include a secure subsystem that can enable control of content use such as displaying, encrypting, decrypting, printing, copying, saving, extracting, embedding, distributing, auditing usage, etc. The secure subsystem in the preferred embodiment comprises one or more "protected processing environments", one or more secure databases, and secure "component assemblies" and other items and processes that need to be kept secured. VDE can, for example, securely control electronic currency, payments, and/or credit management (including electronic credit and/or currency receipt, disbursement, encumbering, and/or allocation) using such a "secure subsystem."

VDE provides a secure, distributed electronic transaction management system for controlling the distribution and/or other usage of electronically provided and/or stored information. VDE

controls auditing and reporting of electronic content and/or
appliance usage. Users of VDE may include content creators
who apply content usage, usage reporting, and/or usage payment
related control information to electronic content and/or
5 appliances for users such as end-user organizations, individuals,
and content and/or appliance distributors. VDE also securely
supports the payment of money owed (including money owed for
content and/or appliance usage) by one or more parties to one or
more other parties, in the form of electronic credit and/or
10 currency.

Electronic appliances under control of VDE represent VDE
'nodes' that securely process and control; distributed electronic
information and/or appliance usage, control information
15 formulation, and related transactions. VDE can securely
manage the integration of control information provided by two or
more parties. As a result, VDE can construct an electronic
agreement between VDE participants that represent a
"negotiation" between, the control requirements of, two or more
20 parties and enacts terms and conditions of a resulting
agreement. VDE ensures the rights of each party to an
electronic agreement regarding a wide range of electronic
activities related to electronic information and/or appliance
usage.

Through use of VDE's control system, traditional content providers and users can create electronic relationships that reflect traditional, non-electronic relationships. They can shape and modify commercial relationships to accommodate the evolving needs of, and agreements among, themselves. VDE does not require electronic content providers and users to modify their business practices and personal preferences to conform to a metering and control application program that supports limited, largely fixed functionality. Furthermore, VDE permits participants to develop business models not feasible with non-electronic commerce, for example, involving detailed reporting of content usage information, large numbers of distinct transactions at hitherto infeasibly low price points, "pass-along" control information that is enforced without involvement or advance knowledge of the participants, etc.

The present invention allows content providers and users to formulate their transaction environment to accommodate:

- (1) desired content models, content control models, and content usage information pathways,
- (2) a complete range of electronic media and distribution means,

- 5
- (3) a broad range of pricing, payment, and auditing strategies,
- (4) very flexible privacy and/or reporting models,
- (5) practical and effective security architectures, and
- 10 (6) other administrative procedures that together with steps (1) through (5) can enable most "real world" electronic commerce and data security models, including models unique to the electronic world.

VDE's transaction management capabilities can enforce:

- 15 (1) privacy rights of users related to information regarding their usage of electronic information and/or appliances,
- 20 (2) societal policy such as laws that protect rights of content users or require the collection of taxes derived from electronic transaction revenue, and

- (3) the proprietary and/or other rights of parties related to ownership of, distribution of, and/or other commercial rights related to, electronic information.

5 VDE can support "real" commerce in an electronic form, that is the progressive creation of commercial relationships that form, over time, a network of interrelated agreements representing a value chain business model. This is achieved in part by enabling content control information to develop through
10 the interaction of (negotiation between) securely created and independently submitted sets of content and/or appliance control information. Different sets of content and/or appliance control information can be submitted by different parties in an electronic business value chain enabled by the present invention. These
15 parties create control information sets through the use of their respective VDE installations. Independently, securely deliverable, component based control information allows efficient interaction among control information sets supplied by different parties.

20

VDE permits multiple, separate electronic arrangements to be formed between subsets of parties in a VDE supported electronic value chain model. These multiple agreements together comprise a VDE value chain "extended" agreement.

VDE allows such constituent electronic agreements, and therefore overall VDE extended agreements, to evolve and reshape over time as additional VDE participants become involved in VDE content and/or appliance control information handling. VDE electronic agreements may also be extended as new control information is submitted by existing participants. With VDE, electronic commerce participants are free to structure and restructure their electronic commerce business activities and relationships. As a result, the present invention allows a competitive electronic commerce marketplace to develop since the use of VDE enables different, widely varying business models using the same or shared content.

A significant facet of the present invention's ability to broadly support electronic commerce is its ability to securely manage independently delivered VDE component objects containing control information (normally in the form of VDE objects containing one or more methods, data, or load module VDE components). This independently delivered control information can be integrated with senior and other pre-existing content control information to securely form derived control information using the negotiation mechanisms of the present invention. All requirements specified by this derived control information must be satisfied before VDE controlled content can

be accessed or otherwise used. This means that, for example, all load modules and any mediating data which are listed by the derived control information as required must be available and securely perform their required function. In combination with
5 other aspects of the present invention, securely, independently delivered control components allow electronic commerce participants to freely stipulate their business requirements and trade offs. As a result, much as with traditional, non-electronic commerce, the present invention allows electronic commerce
10 (through a progressive stipulation of various control requirements by VDE participants) to evolve into forms of business that are the most efficient, competitive and useful.

VDE provides capabilities that rationalize the support of
15 electronic commerce and electronic transaction management. This rationalization stems from the reusability of control structures and user interfaces for a wide variety of transaction management related activities. As a result, content usage control, data security, information auditing, and electronic
20 financial activities, can be supported with tools that are reusable, convenient, consistent, and familiar. In addition, a rational approach—a transaction/distribution control standard—allows all participants in VDE the same foundation set of hardware control and security, authoring, administration,

and management tools to support widely varying types of information, business market model, and/or personal objectives.

Employing VDE as a general purpose electronic transaction/distribution control system allows users to maintain a single transaction management control arrangement on each of their computers, networks, communication nodes, and/or other electronic appliances. Such a general purpose system can serve the needs of many electronic transaction management applications without requiring distinct, different installations for different purposes. As a result, users of VDE can avoid the confusion and expense and other inefficiencies of different, limited purpose transaction control applications for each different content and/or business model. For example, VDE allows content creators to use the same VDE foundation control arrangement for both content authoring and for licensing content from other content creators for inclusion into their products or for other use. Clearinghouses, distributors, content creators, and other VDE users can all interact, both with the applications running on their VDE installations, and with each other, in an entirely consistent manner, using and reusing (largely transparently) the same distributed tools, mechanisms, and consistent user interfaces, regardless of the type of VDE activity.

VDE prevents many forms of unauthorized use of electronic information, by controlling and auditing (and other administration of use) electronically stored and/or disseminated information. This includes, for example, commercially distributed content, electronic currency, electronic credit, business transactions (such as EDI), confidential communications, and the like. VDE can further be used to enable commercially provided electronic content to be made available to users in user defined portions, rather than constraining the user to use portions of content that were "predetermined" by a content creator and/or other provider for billing purposes.

VDE, for example, can employ:

- (1) Secure metering means for budgeting and/or auditing electronic content and/or appliance usage;
- (2) Secure flexible means for enabling compensation and/or billing rates for content and/or appliance usage, including electronic credit and/or currency mechanisms for payment means;

- 5
- (3) Secure distributed database means for storing control and usage related information (and employing validated compartmentalization and tagging schemes);
- (4) Secure electronic appliance control means;
- 10
- (5) A distributed, secure, "virtual black box" comprised of nodes located at every user (including VDE content container creators, other content providers, client users, and recipients of secure VDE content usage information) site. The nodes of said virtual black box normally include a secure subsystem having at least one secure hardware element (a semiconductor element or other hardware module
- 15
- for securely executing VDE control processes), said secure subsystems being distributed at nodes along a pathway of information storage, distribution, payment, usage, and/or auditing. In some
- 20
- embodiments, the functions of said hardware element, for certain or all nodes, may be performed by software, for example, in host processing environments of electronic appliances;

- (6) Encryption and decryption means;
- 5 (7) Secure communications means employing authentication, digital signaturing, and encrypted transmissions. The secure subsystems at said user nodes utilize a protocol that establishes and authenticates each node's and/or participant's identity, and establishes one or more secure host-to-host encryption keys for communications between the secure subsystems; and
- 10 (8) Secure control means that can allow each VDE installation to perform VDE content authoring (placing content into VDE containers with associated control information), content distribution, and content usage; as well as clearinghouse and other administrative and analysis activities employing content usage information.
- 15

20 VDE may be used to migrate most non-electronic, traditional information delivery models (including entertainment, reference materials, catalog shopping, etc.) into an adequately secure digital distribution and usage management

and payment context. The distribution and financial pathways managed by a VDE arrangement may include:

- 5 ● content creator(s),
- distributor(s),
- redistributor(s),
- client administrator(s),
- client user(s),
- financial and/or other clearinghouse(s),
- 10 ● and/or government agencies.

These distribution and financial pathways may also include:

- advertisers,
- 15 ● market survey organizations, and/or
- other parties interested in the user usage of
 information securely delivered and/or stored using
 VDE.

20 Normally, participants in a VDE arrangement will employ the same secure VDE foundation. Alternate embodiments support VDE arrangements employing differing VDE foundations. Such alternate embodiments may employ procedures to ensure certain interoperability requirements are met.

Secure VDE hardware (also known as SPUs for Secure Processing Units), or VDE installations that use software to substitute for, or complement, said hardware (provided by Host Processing Environments (HPEs)), operate in conjunction with

5 secure communications, systems integration software, and distributed software control information and support structures, to achieve the electronic contract/rights protection environment of the present invention. Together, these VDE components

10 comprise a secure, virtual, distributed content and/or appliance control, auditing (and other administration), reporting, and payment environment. In some embodiments and where commercially acceptable, certain VDE participants, such as clearinghouses that normally maintain sufficiently physically

15 secure non-VDE processing environments, may be allowed to employ HPEs rather VDE hardware elements and interoperate, for example, with VDE end-users and content providers. VDE components together comprise a configurable, consistent, secure and "trusted" architecture for distributed, asynchronous control of electronic content and/or appliance usage. VDE supports a

20 "universe wide" environment for electronic content delivery, broad dissemination, usage reporting, and usage related payment activities.

VDE provides generalized configurability. This results, in part, from decomposition of generalized requirements for supporting electronic commerce and data security into a broad range of constituent "atomic" and higher level components (such as load modules, data elements, and methods) that may be variously aggregated together to form control methods for electronic commerce applications, commercial electronic agreements, and data security arrangements. VDE provides a secure operating environment employing VDE foundation elements along with secure independently deliverable VDE components that enable electronic commerce models and relationships to develop. VDE specifically supports the unfolding of distribution models in which content providers, over time, can expressly agree to, or allow, subsequent content providers and/or users to participate in shaping the control information for, and consequences of, use of electronic content and/or appliances. A very broad range of the functional attributes important for supporting simple to very complex electronic commerce and data security activities are supported by capabilities of the present invention. As a result, VDE supports most types of electronic information and/or appliance: usage control (including distribution), security, usage auditing, reporting, other administration, and payment arrangements.

VDE, in its preferred embodiment, employs object software technology and uses object technology to form "containers" for delivery of information that is (at least in part) encrypted or otherwise secured. These containers may contain electronic content products or other electronic information and some or all of their associated permissions (control) information. These container objects may be distributed along pathways involving content providers and/or content users. They may be securely moved among nodes of a Virtual Distribution Environment (VDE) arrangement, which nodes operate VDE foundation software and execute control methods to enact electronic information usage control and/or administration models. The containers delivered through use of the preferred embodiment of the present invention may be employed both for distributing VDE control instructions (information) and/or to encapsulate and electronically distribute content that has been at least partially secured.

Content providers who employ the present invention may include, for example, software application and game publishers, database publishers, cable, television, and radio broadcasters, electronic shopping vendors, and distributors of information in electronic document, book, periodical, e-mail and/or other forms. Corporations, government agencies, and/or individual

“end-users” who act as storers of, and/or distributors of, electronic information, may also be VDE content providers (in a restricted model, a user provides content only to himself and employs VDE to secure his own confidential information against unauthorized use by other parties). Electronic information may include proprietary and/or confidential information for personal or internal organization use, as well as information, such as software applications, documents, entertainment materials, and/or reference information, which may be provided to other parties. Distribution may be by, for example, physical media delivery, broadcast and/or telecommunication means, and in the form of “static” files and/or streams of data. VDE may also be used, for example, for multi-site “real-time” interaction such as teleconferencing, interactive games, or on-line bulletin boards, where restrictions on, and/or auditing of, the use of all or portions of communicated information is enforced.

VDE provides important mechanisms for both enforcing commercial agreements and enabling the protection of privacy rights. VDE can securely deliver information from one party to another concerning the use of commercially distributed electronic content. Even if parties are separated by several “steps” in a chain (pathway) of handling for such content usage information, such information is protected by VDE through encryption and/or

other secure processing. Because of that protection, the accuracy of such information is guaranteed by VDE, and the information can be trusted by all parties to whom it is delivered.

5 Furthermore, VDE guarantees that all parties can trust that such information cannot be received by anyone other than the intended, authorized, party(ies) because it is encrypted such that only an authorized party, or her agents, can decrypt it. Such information may also be derived through a secure VDE process at a previous pathway-of-handling location to produce secure
10 VDE reporting information that is then communicated securely to its intended recipient's VDE secure subsystem. Because VDE can deliver such information securely, parties to an electronic agreement need not trust the accuracy of commercial usage and/or other information delivered through means other than
15 those under control of VDE.

VDE participants in a commercial value chain can be "commercially" confident (that is, sufficiently confident for commercial purposes) that the direct (constituent) and/or
20 "extended" electronic agreements they entered into through the use of VDE can be enforced reliably. These agreements may have both "dynamic" transaction management related aspects, such as content usage control information enforced through budgeting, metering, and/or reporting of electronic information

and/or appliance use, and/or they may include "static" electronic assertions, such as an end-user using the system to assert his or her agreement to pay for services, not to pass to unauthorized parties electronic information derived from usage of content or systems, and/or agreeing to observe copyright laws. Not only can electronically reported transaction related information be trusted under the present invention, but payment may be automated by the passing of payment tokens through a pathway of payment (which may or may not be the same as a pathway for reporting).

Such payment can be contained within a VDE container created automatically by a VDE installation in response to control information (located, in the preferred embodiment, in one or more permissions records) stipulating the "withdrawal" of credit or electronic currency (such as tokens) from an electronic account (for example, an account securely maintained by a user's VDE installation secure subsystem) based upon usage of VDE controlled electronic content and/or appliances (such as governments, financial credit providers, and users).

VDE allows the needs of electronic commerce participants to be served and it can bind such participants together in a universe wide, trusted commercial network that can be secure enough to support very large amounts of commerce. VDE's security and metering secure subsystem core will be present at

all physical locations where VDE related content is (a) assigned
usage related control information (rules and mediating data),
and/or (b) used. This core can perform security and auditing
functions (including metering) that operate within a "virtual
5 black box," a collection of distributed, very secure VDE related
hardware instances that are interconnected by secured
information exchange (for example, telecommunication)
processes and distributed database means. VDE further
includes highly configurable transaction operating system
10 technology, one or more associated libraries of load modules
along with affiliated data, VDE related administration, data
preparation, and analysis applications, as well as system
software designed to enable VDE integration into host
environments and applications. VDE's usage control
15 information, for example, provide for property content and/or
appliance related: usage authorization, usage auditing (which
may include audit reduction), usage billing, usage payment,
privacy filtering, reporting, and security related communication
and encryption techniques.

20

VDE extensively employs methods in the form of software
objects to augment configurability, portability, and security of
the VDE environment. It also employs a software object
architecture for VDE content containers that carries protected

content and may also carry both freely available information (e.g.,
summary, table of contents) and secured content control
information which ensures the performance of control
information. Content control information governs content usage
5 according to criteria set by holders of rights to an object's
contents and/or according to parties who otherwise have rights
associated with distributing such content (such as governments,
financial credit providers, and users).

10 In part, security is enhanced by object methods employed
by the present invention because the encryption schemes used to
protect an object can efficiently be further used to protect the
associated content control information (software control
information and relevant data) from modification. Said object
15 techniques also enhance portability between various computer
and/or other appliance environments because electronic
information in the form of content can be inserted along with (for
example, in the same object container as) content control
information (for said content) to produce a "published" object.
20 As a result, various portions of said control information may be
specifically adapted for different environments, such as for
diverse computer platforms and operating systems, and said
various portions may all be carried by a VDE container.

An objective of VDE is supporting a transaction/distribution control standard. Development of such a standard has many obstacles, given the security requirements and related hardware and communications issues, widely differing environments, information types, types of information usage, business and/or data security goals, varieties of participants, and properties of delivered information. A significant feature of VDE accommodates the many, varying distribution and other transaction variables by, in part, decomposing electronic commerce and data security functions into generalized capability modules executable within a secure hardware SPU and/or corresponding software subsystem and further allowing extensive flexibility in assembling, modifying, and/or replacing, such modules (e.g. load modules and/or methods) in applications run on a VDE installation foundation. This configurability and reconfigurability allows electronic commerce and data security participants to reflect their priorities and requirements through a process of iteratively shaping an evolving extended electronic agreement (electronic control model). This shaping can occur as content control information passes from one VDE participant to another and to the extent allowed by "in place" content control information. This process allows users of VDE to recast existing control

information and/or add new control information as necessary
(including the elimination of no longer required elements).

5 VDE supports trusted (sufficiently secure) electronic
information distribution and usage control models for both
commercial electronic content distribution and data security
applications. It can be configured to meet the diverse
requirements of a network of interrelated participants that may
include content creators, content distributors, client
10 administrators, end users, and/or clearinghouses and/or other
content usage information users. These parties may constitute a
network of participants involved in simple to complex electronic
content dissemination, usage control, usage reporting, and/or
usage payment. Disseminated content may include both
15 originally provided and VDE generated information (such as
content usage information) and content control information may
persist through both chains (one or more pathways) of content
and content control information handling, as well as the direct
usage of content. The configurability provided by the present
20 invention is particularly critical for supporting electronic
commerce, that is enabling businesses to create relationships
and evolve strategies that offer competitive value. Electronic
commerce tools that are not inherently configurable and
interoperable will ultimately fail to produce products (and

services) that meet both basic requirements and evolving needs of most commerce applications.

5 VDE's fundamental configurability will allow a broad range of competitive electronic commerce business models to flourish. It allows business models to be shaped to maximize revenues sources, end-user product value, and operating efficiencies. VDE can be employed to support multiple, differing models, take advantage of new revenue opportunities, and
10 deliver product configurations most desired by users. Electronic commerce technologies that do not, as the present invention does:

- support a broad range of possible, complementary revenue activities,
 - 15 • offer a flexible array of content usage features most desired by customers, and
 - exploit opportunities for operating efficiencies,
- will result in products that are often intrinsically more costly and less appealing and therefore less competitive in the
20 marketplace.

Some of the key factors contributing to the configurability intrinsic to the present invention include:

- 5
- (a) integration into the fundamental control environment of a broad range of electronic appliances through portable API and programming language tools that efficiently support merging of control and auditing capabilities in nearly any electronic appliance environment while maintaining overall system security;
- 10
- (b) modular data structures;
- (c) generic content model;
- (d) general modularity and independence of foundation architectural components;
- 15
- (e) modular security structures;
- (f) variable length and multiple branching chains of control; and
- 20
- (g) independent, modular control structures in the form of executable load modules that can be maintained in one or more libraries, and assembled into control methods and models, and where such model control

schemes can "evolve" as control information passes through the VDE installations of participants of a pathway of VDE content control information handling.

5

Because of the breadth of issues resolved by the present invention, it can provide the emerging "electronic highway" with a single transaction/distribution control system that can, for a very broad range of commercial and data security models, ensure against unauthorized use of confidential and/or proprietary information and commercial electronic transactions. VDE's electronic transaction management mechanisms can enforce the electronic rights and agreements of all parties participating in widely varying business and data security models, and this can be efficiently achieved through a single VDE implementation within each VDE participant's electronic appliance. VDE supports widely varying business and/or data security models that can involve a broad range of participants at various "levels" of VDE content and/or content control information pathways of handling. Different content control and/or auditing models and agreements may be available on the same VDE installation. These models and agreements may control content in relationship to, for example, VDE installations and/or users in general; certain specific users, installations, classes and/or other

10

15

20

groupings of installations and/or users; as well as to electronic content generally on a given installation, to specific properties, property portions, classes and/or other groupings of content.

5 Distribution using VDE may package both the electronic content and control information into the same VDE container, and/or may involve the delivery to an end-user site of different pieces of the same VDE managed property from plural separate remote locations and/or in plural separate VDE content
10 containers and/or employing plural different delivery means. Content control information may be partially or fully delivered separately from its associated content to a user VDE installation in one or more VDE administrative objects. Portions of said control information may be delivered from one or more sources.
15 Control information may also be available for use by access from a user's VDE installation secure sub-system to one or more remote VDE secure sub-systems and/or VDE compatible, certified secure remote locations. VDE control processes such as metering, budgeting, decrypting and/or fingerprinting, may as
20 relates to a certain user content usage activity, be performed in a user's local VDE installation secure subsystem, or said processes may be divided amongst plural secure subsystems which may be located in the same user VDE installations and/or in a network server and in the user installation. For example, a local VDE

installation may perform decryption and save any, or all of, usage metering information related to content and/or electronic appliance usage at such user installation could be performed at the server employing secure (e.g., encrypted) communications
5 between said secure subsystems. Said server location may also be used for near real time, frequent, or more periodic secure receipt of content usage information from said user installation, with, for example, metered information being maintained only temporarily at a local user installation.

10

Delivery means for VDE managed content may include electronic data storage means such as optical disks for delivering one portion of said information and broadcasting and/or telecommunicating means for other portions of said information.

15

Electronic data storage means may include magnetic media, optical media, combined magneto-optical systems, flash RAM memory, bubble memory, and/or other memory storage means such as huge capacity optical storage systems employing holographic, frequency, and/or polarity data storage techniques.

20

Data storage means may also employ layered disc techniques, such as the use of generally transparent and/or translucent materials that pass light through layers of data carrying discs which themselves are physically packaged together as one

thicker disc. Data carrying locations on such discs may be, at least in part, opaque.

5 VDE supports a general purpose foundation for secure transaction management, including usage control, auditing, reporting, and/or payment. This general purpose foundation is called "VDE Functions" ("VDEFs"). VDE also supports a collection of "atomic" application elements (e.g., load modules) that can be selectively aggregated together to form various
10 VDEF capabilities called control methods and which serve as VDEF applications and operating system functions. When a host operating environment of an electronic appliance includes VDEF capabilities, it is called a "Rights Operating System" (ROS). VDEF load modules, associated data, and methods form a body of
15 information that for the purposes of the present invention are called "control information." VDEF control information may be specifically associated with one or more pieces of electronic content and/or it may be employed as a general component of the operating system capabilities of a VDE installation.

20

VDEF transaction control elements reflect and enact content specific and/or more generalized administrative (for example, general operating system) control information. VDEF capabilities which can generally take the form of applications

(application models) that have more or less configurability which can be shaped by VDE participants, through the use, for example, of VDE templates, to employ specific capabilities, along, for example, with capability parameter data to reflect the

5 elements of one or more express electronic agreements between VDE participants in regards to the use of electronic content such as commercially distributed products. These control capabilities manage the use of, and/or auditing of use of, electronic content, as well as reporting information based upon content use, and any

10 payment for said use. VDEF capabilities may "evolve" to reflect the requirements of one or more successive parties who receive or otherwise contribute to a given set of control information. Frequently, for a VDE application for a given content model (such as distribution of entertainment on CD-ROM, content

15 delivery from an Internet repository, or electronic catalog shopping and advertising, or some combination of the above) participants would be able to securely select from amongst available, alternative control methods and apply related parameter data, wherein such selection of control method and/or

20 submission of data would constitute their "contribution" of control information. Alternatively, or in addition, certain control methods that have been expressly certified as securely interoperable and compatible with said application may be independently submitted by a participant as part of such a

contribution. In the most general example, a generally certified load module (certified for a given VDE arrangement and/or content class) may be used with many or any VDE application that operates in nodes of said arrangement. These parties, to the extent they are allowed, can independently and securely add, delete, and/or otherwise modify the specification of load modules and methods, as well as add, delete or otherwise modify related information.

10 Normally the party who creates a VDE content container defines the general nature of the VDEF capabilities that will and/or may apply to certain electronic information. A VDE content container is an object that contains both content (for example, commercially distributed electronic information products such as computer software programs, movies, electronic publications or reference materials, etc.) and certain control information related to the use of the object's content. A creating party may make a VDE container available to other parties. Control information delivered by, and/or otherwise available for use with, VDE content containers comprise (for commercial content distribution purposes) VDEF control capabilities (and any associated parameter data) for electronic content. These capabilities may constitute one or more "proposed" electronic agreements (and/or agreement functions available for selection

and/or use with parameter data) that manage the use and/or the consequences of use of such content and which can enact the terms and conditions of agreements involving multiple parties and their various rights and obligations.

5

A VDE electronic agreement may be explicit, through a user interface acceptance by one or more parties, for example by a "junior" party who has received control information from a "senior" party, or it may be a process amongst equal parties who individually assert their agreement. Agreement may also result from an automated electronic process during which terms and conditions are "evaluated" by certain VDE participant control information that assesses whether certain other electronic terms and conditions attached to content and/or submitted by another party are acceptable (do not violate acceptable control information criteria). Such an evaluation process may be quite simple, for example a comparison to ensure compatibility between a portion of, or all senior, control terms and conditions in a table of terms and conditions and the submitted control information of a subsequent participant in a pathway of content control information handling, or it may be a more elaborate process that evaluates the potential outcome of, and/or implements a negotiation process between, two or more sets of control information submitted by two or more parties. VDE also

10

15

20

accommodates a semi-automated process during which one or more VDE participants directly, through user interface means, resolve "disagreements" between control information sets by accepting and/or proposing certain control information that may
5 be acceptable to control information representing one or more other parties interests and/or responds to certain user interface queries for selection of certain alternative choices and/or for certain parameter information, the responses being adopted if acceptable to applicable senior control information.

10

When another party (other than the first applier of rules), perhaps through a negotiation process, accepts, and/or adds to and/or otherwise modifies, "in place" content control information, a VDE agreement between two or more parties related to the use
15 of such electronic content may be created (so long as any modifications are consistent with senior control information). Acceptance of terms and conditions related to certain electronic content may be direct and express, or it may be implicit as a result of use of content (depending, for example, on legal
20 requirements, previous exposure to such terms and conditions, and requirements of in place control information).

VDEF capabilities may be employed, and a VDE agreement may be entered into, by a plurality of parties without

the VDEF capabilities being directly associated with the controlling of certain, specific electronic information. For example, certain one or more VDEF capabilities may be present at a VDE installation, and certain VDE agreements may have
5 been entered into during the registration process for a content distribution application, to be used by such installation for securely controlling VDE content usage, auditing, reporting and/or payment. Similarly, a specific VDE participant may enter into a VDE user agreement with a VDE content or electronic
10 appliance provider when the user and/or her appliance register with such provider as a VDE installation and/or user. In such events, VDEF in place control information available to the user VDE installation may require that certain VDEF methods are employed, for example in a certain sequence, in order to be able
15 to use all and/or certain classes, of electronic content and/or VDE applications.

VDE ensures that certain prerequisites necessary for a given transaction to occur are met. This includes the secure
20 execution of any required load modules and the availability of any required, associated data. For example, required load modules and data (e.g. in the form of a method) might specify that sufficient credit from an authorized source must be confirmed as available. It might further require certain one or

more load modules execute as processes at an appropriate time to ensure that such credit will be used in order to pay for user use of the content. A certain content provider might, for example, require metering the number of copies made for

5 distribution to employees of a given software program (a portion of the program might be maintained in encrypted form and require the presence of a VDE installation to run). This would require the execution of a metering method for copying of the property each time a copy was made for another employee. This

10 same provider might also charge fees based on the total number of different properties licensed from them by the user and a metering history of their licensing of properties might be required to maintain this information.

15 VDE provides organization, community, and/or universe wide secure environments whose integrity is assured by processes securely controlled in VDE participant user installations (nodes). VDE installations, in the preferred embodiment, may include both software and tamper resistant

20 hardware semiconductor elements. Such a semiconductor arrangement comprises, at least in part, special purpose circuitry that has been designed to protect against tampering with, or unauthorized observation of, the information and functions used in performing the VDE's control functions. The special purpose

secure circuitry provided by the present invention includes at least one of: a dedicated semiconductor arrangement known as a Secure Processing Unit (SPU) and/or a standard microprocessor, microcontroller, and/or other processing logic that accommodates
5 the requirements of the present invention and functions as an SPU. VDE's secure hardware may be found incorporated into, for example, a fax/modem chip or chip pack, I/O controller, video display controller, and/or other available digital processing arrangements. It is anticipated that portions of the present
10 invention's VDE secure hardware capabilities may ultimately be standard design elements of central processing units (CPUs) for computers and various other electronic devices.

Designing VDE capabilities into one or more standard
15 microprocessor, microcontroller and/or other digital processing components may materially reduce VDE related hardware costs by employing the same hardware resources for both the transaction management uses contemplated by the present invention and for other, host electronic appliance functions. This
20 means that a VDE SPU can employ (share) circuitry elements of a "standard" CPU. For example, if a "standard" processor can operate in protected mode and can execute VDE related instructions as a protected activity, then such an embodiment may provide sufficient hardware security for a variety of

applications and the expense of a special purpose processor might be avoided. Under one preferred embodiment of the present invention, certain memory (e.g., RAM, ROM, NVRAM) is maintained during VDE related instruction processing in a
5 protected mode (for example, as supported by protected mode microprocessors). This memory is located in the same package as the processing logic (e.g. processor). Desirably, the packaging and memory of such a processor would be designed using security techniques that enhance its resistance to tampering.

10

The degree of overall security of the VDE system is primarily dependent on the degree of tamper resistance and concealment of VDE control process execution and related data storage activities. Employing special purpose semiconductor
15 packaging techniques can significantly contribute to the degree of security. Concealment and tamper-resistance in semiconductor memory (e.g., RAM, ROM, NVRAM) can be achieved, in part, by employing such memory within an SPU package, by encrypting data before it is sent to external memory
20 (such as an external RAM package) and decrypting encrypted data within the CPU/RAM package before it is executed. This process is used for important VDE related data when such data is stored on unprotected media, for example, standard host storage, such as random access memory, mass storage, etc. In

that event, a VDE SPU would encrypt data that results from a secure VDE execution before such data was stored in external memory.

5 **Summary of Some Important Features Provided by VDE in Accordance With the Present Invention**

VDE employs a variety of capabilities that serve as a foundation for a general purpose, sufficiently secure distributed electronic commerce solution. VDE enables an electronic
10 commerce marketplace that supports divergent, competitive business partnerships, agreements, and evolving overall business models. For example, VDE includes features that:

15 “sufficiently” impede unauthorized and/or uncompensated use of electronic information and/or appliances through the use of secure communication, storage, and transaction management technologies. VDE supports a model
20 wide, distributed security implementation which creates a single secure “virtual” transaction processing and information storage environment. VDE enables distributed VDE installations to securely store and communicate information and remotely control the execution processes and the

character of use of electronic information at other
VDE installations and in a wide variety of ways;

- support low-cost, efficient, and effective security architectures for transaction control, auditing, reporting, and related communications and information storage. VDE may employ tagging related security techniques, the time-ageing of encryption keys, the compartmentalization of both stored control information (including differentially tagging such stored information to ensure against substitution and tampering) and distributed content (to, for many content applications, employ one or more content encryption keys that are unique to the specific VDE installation and/or user), private key techniques such as triple DES to encrypt content, public key techniques such as RSA to protect communications and to provide the benefits of digital signature and authentication to securely bind together the nodes of a VDE arrangement, secure processing of important transaction management executable code, and a combining of a small amount of highly secure, hardware protected storage space with a much larger "exposed" mass media storage

space storing secured (normally encrypted and tagged) control and audit information. VDE employs special purpose hardware distributed throughout some or all locations of a VDE implementation: a) said hardware controlling important elements of: content preparation (such as causing such content to be placed in a VDE content container and associating content control information with said content), content and/or electronic appliance usage auditing, content usage analysis, as well as content usage control; and b) said hardware having been designed to securely handle processing load module control activities, wherein said control processing activities may involve a sequence of required control factors;

- support dynamic user selection of information subsets of a VDE electronic information product (VDE controlled content). This contrasts with the constraints of having to use a few high level individual, pre-defined content provider information increments such as being required to select a whole information product or product section in order to acquire or otherwise use a portion of such product or

section. VDE supports metering and usage control over a variety of increments (including "atomic" increments, and combinations of different increment types) that are selected ad hoc by a user and

5 represent a collection of pre-identified one or more increments (such as one or more blocks of a preidentified nature, e.g., bytes, images, logically related blocks) that form a generally arbitrary, but logical to a user, content "deliverable." VDE control

10 information (including budgeting, pricing and metering) can be configured so that it can specifically apply, as appropriate, to ad hoc selection of different, unanticipated variable user selected aggregations of information increments and pricing

15 levels can be, at least in part, based on quantities and/or nature of mixed increment selections (for example, a certain quantity of certain text could mean associated images might be discounted by 15%; a greater quantity of text in the "mixed"

20 increment selection might mean the images are discounted 20%). Such user selected aggregated information increments can reflect the actual requirements of a user for information and is more flexible than being limited to a single, or a few, high

level, (e.g. product, document, database record).
predetermined increments. Such high level
increments may include quantities of information
not desired by the user and as a result be more
5 costly than the subset of information needed by the
user if such a subset was available. In sum, the
present invention allows information contained in
electronic information products to be supplied
according to user specification. Tailoring to user
10 specification allows the present invention to provide
the greatest value to users, which in turn will
generate the greatest amount of electronic commerce
activity. The user, for example, would be able to
define an aggregation of content derived from
15 various portions of an available content product, but
which, as a deliverable for use by the user, is an
entirely unique aggregated increment. The user
may, for example, select certain numbers of bytes of
information from various portions of an information
20 product, such as a reference work, and copy them to
disc in unencrypted form and be billed based on
total number of bytes plus a surcharge on the
number of "articles" that provided the bytes. A
content provider might reasonably charge less for

such a user defined information increment since the user does not require all of the content from all of the articles that contained desired information. This process of defining a user desired information

5 increment may involve artificial intelligence database search tools that contribute to the location of the most relevant portions of information from an information product and cause the automatic display to the user of information describing search criteria

10 hits for user selection or the automatic extraction and delivery of such portions to the user. VDE further supports a wide variety of predefined increment types including:

- bytes,
- 15 • images,
- content over time for audio or video, or any other increment that can be identified by content provider data mapping efforts, such as:
- sentences,
- 20 • paragraphs,
- articles,
- database records, and
- byte offsets representing increments of logically related information.

VDE supports as many simultaneous predefined increment types as may be practical for a given type of content and business model.

- 5
- securely store at a user's site potentially highly detailed information reflective of a user's usage of a variety of different content segment types and employing both inexpensive "exposed" host mass storage for maintaining detailed information in the form of encrypted data and maintaining summary information for security testing in highly secure special purpose VDE installation nonvolatile memory (if available).
- 10
- 15
- support trusted chain of handling capabilities for pathways of distributed electronic information and/or for content usage related information. Such chains may extend, for example, from a content creator, to a distributor, a redistributor, a client user, and then may provide a pathway for securely reporting the same and/or differing usage information to one or more auditors, such as to one or more independent clearinghouses and then back to the content providers, including content creators.
- 20

5 The same and/or different pathways employed for
certain content handling, and related content control
information and reporting information handling,
may also be employed as one or more pathways for
electronic payment handling (payment is
characterized in the present invention as
administrative content) for electronic content and/or
appliance usage. These pathways are used for
conveyance of all or portions of content, and/or
10 content related control information. Content
creators and other providers can specify the
pathways that, partially or fully, must be used to
disseminate commercially distributed property
content, content control information, payment
15 administrative content, and/or associated usage
reporting information. Control information specified
by content providers may also specify which specific
parties must or may (including, for example, a group
of eligible parties from which a selection may be
20 made) handle conveyed information. It may also
specify what transmission means (for example
telecommunication carriers or media types) and
transmission hubs must or may be used.

- support flexible auditing mechanisms, such as employing “bitmap meters,” that achieve a high degree of efficiency of operation and throughput and allow, in a practical manner, the retention and ready recall of information related to previous usage activities and related patterns. This flexibility is adaptable to a wide variety of billing and security control strategies such as:
 - upgrade pricing (e.g. suite purchases),
 - pricing discounts (including quantity discounts),
 - billing related time duration variables such as discounting new purchases based on the timing of past purchases, and
 - security budgets based on quantity of different, logically related units of electronic information used over an interval of time.

Use of bitmap meters (including “regular” and “wide” bitmap meters) to record usage and/or purchase of information, in conjunction with other elements of the preferred embodiment of the present invention, uniquely supports efficient maintenance of usage history for: (a) rental, (b) flat fee licensing

or purchase, (c) licensing or purchase discounts based upon historical usage variables, and (d) reporting to users in a manner enabling users to determine whether a certain item was acquired, or
5 acquired within a certain time period (without requiring the use of conventional database mechanisms, which are highly inefficient for these applications). Bitmap meter methods record activities associated with electronic appliances,
10 properties, objects, or portions thereof, and/or administrative activities that are independent of specific properties, objects, etc., performed by a user and/or electronic appliance such that a content and/or appliance provider and/or controller of an
15 administrative activity can determine whether a certain activity has occurred at some point, or during a certain period, in the past (for example, certain use of a commercial electronic content product and/or appliance). Such determinations can
20 then be used as part of pricing and/or control strategies of a content and/or appliance provider, and/or controller of an administrative activity. For example, the content provider may choose to charge only once for access to a portion of a property,

regardless of the number of times that portion of the property is accessed by a user.

- 5 • support “launchable” content, that is content that can be provided by a content provider to an end-user, who can then copy or pass along the content to other end-user parties without requiring the direct participation of a content provider to register and/or otherwise initialize the content for 10 use. This content goes “out of (the traditional distribution) channel” in the form of a “traveling object.” Traveling objects are containers that securely carry at least some permissions information and/or methods that are required for their use (such 15 methods need not be carried by traveling objects if the required methods will be available at, or directly available to, a destination VDE installation).
Certain travelling objects may be used at some or all VDE installations of a given VDE arrangement since 20 they can make available the content control information necessary for content use without requiring the involvement of a commercial VDE value chain participant or data security administrator (e.g. a control officer or network

administrator). As long as traveling object control information requirements are available at the user VDE installation secure subsystem (such as the presence of a sufficient quantity of financial credit from an authorized credit provider), at least some travelling object content may be used by a receiving party without the need to establish a connection with a remote VDE authority (until, for example, budgets are exhausted or a time content usage reporting interval has occurred). Traveling objects can travel "out-of-channel," allowing, for example, a user to give a copy of a traveling object whose content is a software program, a movie or a game, to a neighbor, the neighbor being able to use the traveling object if appropriate credit (e.g. an electronic clearinghouse account from a clearinghouse such as VISA or AT&T) is available. Similarly, electronic information that is generally available on an Internet, or a similar network, repository might be provided in the form of a traveling object that can be downloaded and subsequently copied by the initial downloader and then passed along to other parties who may pass the object on to additional parties.

- 5 • provide very flexible and extensible user
 identification according to individuals, installations,
 by groups such as classes, and by function and
 hierarchical identification employing a hierarchy of
10 levels of client identification (for example, client
 organization ID, client department ID, client
 network ID, client project ID, and client employee
 ID, or any appropriate subset of the above).

- 15 • provide a general purpose, secure, component based
 content control and distribution system that
 functions as a foundation transaction operating
 system environment that employs executable code
 pieces crafted for transaction control and auditing.
20 These code pieces can be reused to optimize
 efficiency in creation and operation of trusted,
 distributed transaction management arrangements.
 VDE supports providing such executable code in the
 form of "atomic" load modules and associated data.
 Many such load modules are inherently
 configurable, aggregatable, portable, and extensible
 and singularly, or in combination (along with
 associated data), run as control methods under the
 VDE transaction operating environment. VDE can

satisfy the requirements of widely differing
electronic commerce and data security applications
by, in part, employing this general purpose
transaction management foundation to securely
5 process VDE transaction related control methods.
Control methods are created primarily through the
use of one or more of said executable, reusable load
module code pieces (normally in the form of
executable object components) and associated data.
10 The component nature of control methods allows the
present invention to efficiently operate as a highly
configurable content control system. Under the
present invention, content control models can be
iteratively and asynchronously shaped, and
15 otherwise updated to accommodate the needs of
VDE participants to the extent that such shaping
and otherwise updating conforms to constraints
applied by a VDE application, if any (e.g., whether
new component assemblies are accepted and, if so,
20 what certification requirements exist for such
component assemblies or whether any or certain
participants may shape any or certain control
information by selection amongst optional control
information (permissions record) control methods.

This iterative (or concurrent) multiple participant process occurs as a result of the submission and use of secure, control information components (executable code such as load modules and/or methods, and/or associated data). These components may be contributed independently by secure communication between each control information influencing VDE participant's VDE installation and may require certification for use with a given application, where such certification was provided by a certification service manager for the VDE arrangement who ensures secure interoperability and/or reliability (e.g., bug control resulting from interaction) between appliances and submitted control methods. The transaction management control functions of a VDE electronic appliance transaction operating environment interact with non-secure transaction management operating system functions to properly direct transaction processes and data related to electronic information security, usage control, auditing, and usage reporting. VDE provides the capability to manages resources related to secure VDE content

and/or appliance control information execution and data storage.

- 5 ● facilitate creation of application and/or system
functionality under VDE and to facilitate integration
into electronic appliance environments of load
modules and methods created under the present
invention. To achieve this, VDE employs an
Application Programmer's Interface (API) and/or a
10 transaction operating system (such as a ROS)
programming language with incorporated functions,
both of which support the use of capabilities and can
be used to efficiently and tightly integrate VDE
functionality into commercial and user applications.
15

- 20 ● support user interaction through: (a) "Pop-Up"
applications which, for example, provide messages to
users and enable users to take specific actions such
as approving a transaction, (b) stand-alone VDE
applications that provide administrative
environments for user activities such as: end-user
preference specifications for limiting the price per
transaction, unit of time, and/or session, for

accessing history information concerning previous transactions, for reviewing financial information such as budgets, expenditures (e.g. detailed and/or summary) and usage analysis information, and (c)

5 VDE aware applications which, as a result of the use of a VDE API and/or a transaction management (for example, ROS based) programming language embeds VDE "awareness" into commercial or

10 internal software (application programs, games, etc.) so that VDE user control information and services are seamlessly integrated into such software and can be directly accessed by a user since the

underlying functionality has been integrated into the commercial software's native design. For

15 example, in a VDE aware word processor application, a user may be able to "print" a document into a VDE content container object, applying specific control information by selecting

from amongst a series of different menu templates for different purposes (for example, a confidential

20 memo template for internal organization purposes may restrict the ability to "keep," that is to make an electronic copy of the memo).

- employ “templates” to ease the process of configuring capabilities of the present invention as they relate to specific industries or businesses. Templates are applications or application add-ons under the present invention. Templates support the efficient specification and/or manipulation of criteria related to specific content types, distribution approaches, pricing mechanisms, user interactions with content and/or administrative activities, and/or the like.
- 5
- 10
- 15
- 20
- Given the very large range of capabilities and configurations supported by the present invention, reducing the range of configuration opportunities to a manageable subset particularly appropriate for a given business model allows the full configurable power of the present invention to be easily employed by “typical” users who would be otherwise burdened with complex programming and/or configuration design responsibilities template applications can also help ensure that VDE related processes are secure and optimally bug free by reducing the risks associated with the contribution of independently developed load modules, including unpredictable aspects of code interaction between independent modules and applications, as well as security risks

associated with possible presence of viruses in such modules. VDE, through the use of templates, reduces typical user configuration responsibilities to an appropriately focused set of activities including

5 selection of method types (e.g. functionality) through menu choices such as multiple choice, icon selection, and/or prompting for method parameter data (such as identification information, prices, budget limits, dates, periods of time, access rights to specific

10 content, etc.) that supply appropriate and/or necessary data for control information purposes. By limiting the typical (non-programming) user to a limited subset of configuration activities whose general configuration environment (template) has

15 been preset to reflect general requirements corresponding to that user, or a content or other business model can very substantially limit difficulties associated with content containerization (including placing initial control information on

20 content), distribution, client administration, electronic agreement implementation, end-user interaction, and clearinghouse activities, including associated interoperability problems (such as conflicts resulting from security, operating system,

and/or certification incompatibilities). Use of appropriate VDE templates can assure users that their activities related to content VDE containerization, contribution of other control information, communications, encryption techniques and/or keys, etc. will be in compliance with specifications for their distributed VDE arrangement. VDE templates constitute preset configurations that can normally be reconfigurable to allow for new and/or modified templates that reflect adaptation into new industries as they evolve or to reflect the evolution or other change of an existing industry. For example, the template concept may be used to provide individual, overall frameworks for organizations and individuals that create, modify, market, distribute, consume, and/or otherwise use movies, audio recordings and live performances, magazines, telephony based retail sales, catalogs, computer software, information data bases, multimedia, commercial communications, advertisements, market surveys, infomercials, games, CAD/CAM services for numerically controlled machines, and the like. As the context surrounding these templates changes or evolves,

template applications provided under the present invention may be modified to meet these changes for broad use, or for more focused activities. A given VDE participant may have a plurality of templates available for different tasks. A party that places content in its initial VDE container may have a variety of different, configurable templates depending on the type of content and/or business model related to the content. An end-user may have different configurable templates that can be applied to different document types (e-mail, secure internal documents, database records, etc.) and/or subsets of users (applying differing general sets of control information to different bodies of users, for example, selecting a list of users who may, under certain preset criteria, use a certain document). Of course, templates may, under certain circumstances have fixed control information and not provide for user selections or parameter data entry.

- support plural, different control models regulating the use and/or auditing of either the same specific copy of electronic information content and/or differently regulating different copies (occurrences)

of the same electronic information content.

Differing models for billing, auditing, and security can be applied to the same piece of electronic information content and such differing sets of control information may employ, for control purposes, the same, or differing, granularities of electronic information control increments. This includes supporting variable control information for budgeting and auditing usage as applied to a variety of predefined increments of electronic information, including employing a variety of different budgets and/or metering increments for a given electronic information deliverable for: billing units of measure, credit limit, security budget limit and security content metering increments, and/or market surveying and customer profiling content metering increments. For example, a CD-ROM disk with a database of scientific articles might be in part billed according to a formula based on the number of bytes decrypted, number of articles containing said bytes decrypted, while a security budget might limit the use of said database to no more than 5% of the database per month for users on the wide area network it is installed on.

- provide mechanisms to persistently maintain trusted content usage and reporting control information through both a sufficiently secure chain of handling of content and content control information and through various forms of usage of such content wherein said persistence of control may survive such use. Persistence of control includes the ability to extract information from a VDE container object by creating a new container whose contents are at least in part secured and that contains both the extracted content and at least a portion of the control information which control information of the original container and/or are at least in part produced by control information of the original container for this purpose and/or VDE installation control information stipulates should persist and/or control usage of content in the newly formed container. Such control information can continue to manage usage of container content if the container is "embedded" into another VDE managed object, such as an object which contains plural embedded VDE containers, each of which contains content derived (extracted) from a different source.

- enables users, other value chain participants (such as clearinghouses and government agencies), and/or user organizations, to specify preferences or requirements related to their use of electronic content and/or appliances. Content users, such as end-user customers using commercially distributed content (games, information resources, software programs, etc.), can define, if allowed by senior control information, budgets, and/or other control information, to manage their own internal use of content. Uses include, for example, a user setting a limit on the price for electronic documents that the user is willing to pay without prior express user authorization, and the user establishing the character of metering information he or she is willing to allow to be collected (privacy protection). This includes providing the means for content users to protect the privacy of information derived from their use of a VDE installation and content and/or appliance usage auditing. In particular, VDE can prevent information related to a participant's usage of electronic content from being provided to other parties without the participant's tacit or explicit agreement.

- provide mechanisms that allow control information to “evolve” and be modified according, at least in part, to independently, securely delivered further control information. Said control information may include executable code (e.g., load modules) that has been certified as acceptable (e.g., reliable and trusted) for use with a specific VDE application, class of applications, and/or a VDE distributed arrangement. This modification (evolution) of control information can occur upon content control information (load modules and any associated data) circulating to one or more VDE participants in a pathway of handling of control information, or it may occur upon control information being received from a VDE participant. Handlers in a pathway of handling of content control information, to the extent each is authorized, can establish, modify, and/or contribute to, permission, auditing, payment, and reporting control information related to controlling, analyzing, paying for, and/or reporting usage of, electronic content and/or appliances (for example, as related to usage of VDE controlled property content). Independently delivered (from an independent source which is independent except in

regards to certification), at least in part secure, control information can be employed to securely modify content control information when content control information has flowed from one party to another party in a sequence of VDE content control information handling. This modification employs, for example, one or more VDE component assemblies being securely processed in a VDE secure subsystem. In an alternate embodiment, control information may be modified by a senior party through use of their VDE installation secure sub-system after receiving submitted, at least in part secured, control information from a "junior" party, normally in the form of a VDE administrative object. Control information passing along VDE pathways can represent a mixed control set, in that it may include: control information that persisted through a sequence of control information handlers, other control information that was allowed to be modified, and further control information representing new control information and/or mediating data. Such a control set represents an evolution of control information for disseminated content. In this example the overall content control

set for a VDE content container is "evolving" as it securely (e.g. communicated in encrypted form and using authentication and digital signaturing techniques) passes, at least in part, to a new participant's VDE installation where the proposed control information is securely received and handled. The received control information may be integrated (through use of the receiving parties' VDE installation secure sub-system) with in-place control information through a negotiation process involving both control information sets. For example, the modification, within the secure sub-system of a content provider's VDE installation, of content control information for a certain VDE content container may have occurred as a result of the incorporation of required control information provided by a financial credit provider. Said credit provider may have employed their VDE installation to prepare and securely communicate (directly or indirectly) said required control information to said content provider. Incorporating said required control information enables a content provider to allow the credit provider's credit to be employed by a content end-user to compensate for the end-user's

use of VDE controlled content and/or appliances, so long as said end-user has a credit account with said financial credit provider and said credit account has sufficient credit available. Similarly, control

5 information requiring the payment of taxes and/or the provision of revenue information resulting from electronic commerce activities may be securely received by a content provider. This control

10 information may be received, for example, from a government agency. Content providers might be required by law to incorporate such control information into the control information for

15 commercially distributed content and/or services related to appliance usage. Proposed control information is used to an extent allowed by senior control information and as determined by any negotiation trade-offs that satisfy priorities stipulated by each set (the received set and the

20 proposed set). VDE also accommodates different control schemes specifically applying to different participants (e.g., individual participants and/or participant classes (types)) in a network of VDE content handling participants.

- support multiple simultaneous control models for the same content property and/or property portion. This allows, for example, for concurrent business activities which are dependent on electronic commercial product content distribution, such as acquiring detailed market survey information and/or supporting advertising, both of which can increase revenue and result in lower content costs to users and greater value to content providers. Such control information and/or overall control models may be applied, as determined or allowed by control information, in differing manners to different participants in a pathway of content, reporting, payment, and/or related control information handling. VDE supports applying different content control information to the same and/or different content and/or appliance usage related activities, and/or to different parties in a content and/or appliance usage model, such that different parties (or classes of VDE users, for example) are subject to differing control information managing their use of electronic information content. For example, differing control models based on the category of a user as a distributor of a VDE controlled content

5 object or an end-user of such content may result in
different budgets being applied. Alternatively, for
example, a one distributor may have the right to
distribute a different array of properties than
another distributor (from a common content
collection provided, for example, on optical disc). An
individual, and/or a class or other grouping of
end-users, may have different costs (for example, a
student, senior citizen, and/or poor citizen user of
10 content who may be provided with the same or
differing discounts) than a "typical" content user.

15 • support provider revenue information resulting from
customer use of content and/or appliances, and/or
provider and/or end-user payment of taxes, through
the transfer of credit and/or electronic currency from
said end-user and/or provider to a government
agency, might occur "automatically" as a result of
such received control information causing the
20 generation of a VDE content container whose
content includes customer content usage information
reflecting secure, trusted revenue summary
information and/or detailed user transaction listings
(level of detail might depend, for example on type or

size of transaction—information regarding a bank interest payment to a customer or a transfer of a large (e.g. over \$10,000) might be, by law, automatically reported to the government). Such summary and/or detailed information related to taxable events and/or currency, and/or creditor currency transfer, may be passed along a pathway of reporting and/or payment to the government in a VDE container. Such a container may also be used for other VDE related content usage reporting information.

- support the flowing of content control information through different “branches” of content control information handling so as to accommodate, under the present invention’s preferred embodiment, diverse controlled distributions of VDE controlled content. This allows different parties to employ the same initial electronic content with differing (perhaps competitive) control strategies. In this instance, a party who first placed control information on content can make certain control assumptions and these assumptions would evolve into more specific and/or extensive control

assumptions. These control assumptions can evolve during the branching sequence upon content model participants submitting control information changes, for example, for use in "negotiating" with "in place" content control information. This can result in new or modified content control information and/or it might involve the selection of certain one or more already "in-place" content usage control methods over in-place alternative methods, as well as the submission of relevant control information parameter data. This form of evolution of different control information sets applied to different copies of the same electronic property content and/or appliance results from VDE control information flowing "down" through different branches in an overall pathway of handling and control and being modified differently as it diverges down these different pathway branches. This ability of the present invention to support multiple pathway branches for the flow of both VDE content control information and VDE managed content enables an electronic commerce marketplace which supports diverging, competitive business partnerships, agreements, and evolving overall business models

which can employ the same content properties combined, for example, in differing collections of content representing differing at least in part competitive products.

5

- enable a user to securely extract, through the use of the secure subsystem at the user's VDE installation, at least a portion of the content included within a VDE content container to produce a new, secure object (content container), such that the extracted information is maintained in a continually secure manner through the extraction process. Formation of the new VDE container containing such extracted content shall result in control information consistent with, or specified by, the source VDE content container, and/or local VDE installation secure subsystem as appropriate, content control information. Relevant control information, such as security and administrative information, derived, at least in part, from the parent (source) object's control information, will normally be automatically inserted into a new VDE content container object containing extracted VDE content. This process typically occurs under the control framework of a

10

15

20

parent object and/or VDE installation control
information executing at the user's VDE installation
secure subsystem (with, for example, at least a
portion of this inserted control information being
5 stored securely in encrypted form in one or more
permissions records). In an alternative embodiment,
the derived content control information applied to
extracted content may be in part or whole derived
from, or employ, content control information stored
10 remotely from the VDE installation that performed
the secure extraction such as at a remote server
location. As with the content control information for
most VDE managed content, features of the present
invention allows the content's control information to:

15

- (a) "evolve," for example, the extractor of content
may add new control methods and/or modify
control parameter data, such as VDE
application compliant methods, to the extent
20 allowed by the content's in-place control
information. Such new control information
might specify, for example, who may use at
least a portion of the new object, and/or how
said at least a portion of said extracted

content may be used (e.g. when at least a portion may be used, or what portion or quantity of portions may be used);

- 5
- (b) allow a user to combine additional content with at least a portion of said extracted content, such as material authored by the extractor and/or content (for example, images, video, audio, and/or text) extracted from one or more other VDE container objects for placement directly into the new container;
- 10
- (c) allow a user to securely edit at least a portion of said content while maintaining said content in a secure form within said VDE content container;
- 15
- (d) append extracted content to a pre-existing VDE content container object and attach associated control information -- in these cases, user added information may be secured, e.g., encrypted, in part or as a whole, and may be subject to usage and/or auditing control
- 20

information that differs from the those applied to previously in place object content;

- 5 (e) preserve VDE control over one or more portions of extracted content after various forms of usage of said portions, for example, maintain content in securely stored form while allowing "temporary" on screen display of content or allowing a software program to be maintained in secure form but transiently decrypt any encrypted executing portion of said program (all, or only a portion, of said program may be encrypted to secure the program).
- 10
- 15

Generally, the extraction features of the present invention allow users to aggregate and/or disseminate and/or otherwise use protected electronic content information extracted from content container sources while maintaining secure VDE capabilities thus preserving the rights of providers in said content information after various content usage processes.

20

• support the aggregation of portions of VDE controlled content, such portions being subject to differing VDE content container control information, wherein various of said portions may have been provided by independent, different content providers from one or more different locations remote to the user performing the aggregation. Such aggregation, in the preferred embodiment of the present invention, may involve preserving at least a portion of the control information (e.g., executable code such as load modules) for each of various of said portions by, for example, embedding some or all of such portions individually as VDE content container objects within an overall VDE content container and/or embedding some or all of such portions directly into a VDE content container. In the latter case, content control information of said content container may apply differing control information sets to various of such portions based upon said portions original control information requirements before aggregation. Each of such embedded VDE content containers may have its own control information in the form of one or more permissions records. Alternatively, a negotiation between

control information associated with various aggregated portions of electronic content, may produce a control information set that would govern some or all of the aggregated content portions. The

5 VDE content control information produced by the negotiation may be uniform (such as having the same load modules and/or component assemblies, and/or it may apply differing such content control information to two or more portions that constitute

10 an aggregation of VDE controlled content such as differing metering, budgeting, billing and/or payment models. For example, content usage payment may be automatically made, either through a clearinghouse, or directly, to different content

15 providers for different portions.

- enable flexible metering of, or other collection of information related to, use of electronic content and/or electronic appliances. A feature of the
- 20 present invention enables such flexibility of metering control mechanisms to accommodate a simultaneous, broad array of: (a) different parameters related to electronic information content use; (b) different increment units (bytes, documents,

properties, paragraphs, images, etc.) and/or other organizations of such electronic content; and/or (c) different categories of user and/or VDE installation types, such as client organizations, departments, projects, networks, and/or individual users, etc.

5

This feature of the present invention can be employed for content security, usage analysis (for example, market surveying), and/or compensation based upon the use and/or exposure to VDE managed content. Such metering is a flexible basis for ensuring payment for content royalties, licensing, purchasing, and/or advertising. A feature of the present invention provides for payment means supporting flexible electronic currency and credit mechanisms, including the ability to securely maintain audit trails reflecting information related to use of such currency or credit. VDE supports multiple differing hierarchies of client organization control information wherein an organization client administrator distributes control information specifying the usage rights of departments, users, and/or projects. Likewise, a department (division) network manager can function as a distributor

10

15

20

(budgets, access rights, etc.) for department networks, projects, and/or users, etc.

- 5 • provide scalable, integratable, standardized control means for use on electronic appliances ranging from inexpensive consumer (for example, television set-top appliances) and professional devices (and hand-held PDAs) to servers, mainframes, communication switches, etc. The scalable
10 transaction management/auditing technology of the present invention will result in more efficient and reliable interoperability amongst devices functioning in electronic commerce and/or data security
15 environments. As standardized physical containers have become essential to the shipping of physical goods around the world, allowing these physical containers to universally “fit” unloading equipment, efficiently use truck and train space, and
20 accommodate known arrays of objects (for example, boxes) in an efficient manner, so VDE electronic content containers may, as provided by the present invention, be able to efficiently move electronic information content (such as commercially published properties, electronic currency and credit, and

content audit information), and associated content control information, around the world.

5 Interoperability is fundamental to efficient electronic commerce. The design of the VDE foundation, VDE load modules, and VDE containers, are important features that enable the VDE node operating environment to be compatible with a very broad range of electronic appliances. The ability, for example, for control methods based on load modules to execute in very "small" and inexpensive secure sub-system environments, such as environments with very little read/write memory, while also being able to execute in large memory sub-systems that may be used in more expensive electronic appliances, supports consistency across many machines. This consistent VDE operating environment, including its control structures and container architecture, enables the use of standardized VDE content containers across a broad range of device types and host operating environments. Since VDE capabilities can be seamlessly integrated as extensions, additions, and/or modifications to fundamental capabilities of electronic appliances and host operating systems,

5

10

15

20

VDE containers, content control information, and the VDE foundation will be able to work with many device types and these device types will be able to consistently and efficiently interpret and enforce VDE control information. Through this integration users can also benefit from a transparent interaction with many of the capabilities of VDE. VDE integration with software operating on a host electronic appliance supports a variety of capabilities that would be unavailable or less secure without such integration. Through integration with one or more device applications and/or device operating environments, many capabilities of the present invention can be presented as inherent capabilities of a given electronic appliance, operating system, or appliance application. For example, features of the present invention include:

(a) VDE system software to in part extend and/or modify host operating systems such that they possesses VDE capabilities, such as enabling secure transaction processing and electronic information storage; (b) one or more application programs that in part represent tools associated with VDE operation; and/or (c) code to be integrated into application

5 programs, wherein such code incorporates references
into VDE system software to integrate VDE
capabilities and makes such applications VDE
aware (for example, word processors, database
10 retrieval applications, spreadsheets, multimedia
presentation authoring tools, film editing software,
music editing software such as MIDI applications
and the like, robotics control systems such as those
associated with CAD/CAM environments and NCM
15 software and the like, electronic mail systems,
teleconferencing software, and other data authoring,
creating, handling, and/or usage applications
including combinations of the above). These one or
more features (which may also be implemented in
20 firmware or hardware) may be employed in
conjunction with a VDE node secure hardware
processing capability, such as a microcontroller(s),
microprocessor(s), other CPU(s) or other digital
processing logic.

- employ audit reconciliation and usage pattern
evaluation processes that assess, through certain,
normally network based, transaction processing
reconciliation and threshold checking activities,

whether certain violations of security of a VDE arrangement have occurred. These processes are performed remote to VDE controlled content end-user VDE locations by assessing, for example, purchases, and/or requests, for electronic properties by a given VDE installation. Applications for such reconciliation activities include assessing whether the quantity of remotely delivered VDE controlled content corresponds to the amount of financial credit and/or electronic currency employed for the use of such content. A trusted organization can acquire information from content providers concerning the cost for content provided to a given VDE installation and/or user and compare this cost for content with the credit and/or electronic currency disbursements for that installation and/or user. Inconsistencies in the amount of content delivered versus the amount of disbursement can prove, and/or indicate, depending on the circumstances, whether the local VDE installation has been, at least to some degree, compromised (for example, certain important system security functions, such as breaking encryption for at least some portion of the secure subsystem and/or VDE controlled content by uncovering one or more

keys). Determining whether irregular patterns (e.g. unusually high demand) of content usage, or requests for delivery of certain kinds of VDE controlled information during a certain time period by one or more VDE installations and/or users (including, for example, groups of related users whose aggregate pattern of usage is suspicious) may also be useful in determining whether security at such one or more installations, and/or by such one or more users, has been compromised, particularly when used in combination with an assessment of electronic credit and/or currency provided to one or more VDE users and/or installations, by some or all of their credit and/or currency suppliers, compared with the disbursements made by such users and/or installations.

- support security techniques that materially increase the time required to "break" a system's integrity. This includes using a collection of techniques that minimizes the damage resulting from comprising some aspect of the security features of the present inventions.

- provide a family of authoring, administrative, reporting, payment, and billing tool user applications that comprise components of the present invention's trusted/secure, universe wide, distributed transaction control and administration system. These components support VDE related: object creation (including placing control information on content), secure object distribution and management (including distribution control information, financial related, and other usage analysis), client internal VDE activities administration and control, security management, user interfaces, payment disbursement, and clearinghouse related functions. These components are designed to support highly secure, uniform, consistent, and standardized: electronic commerce and/or data security pathway(s) of handling, reporting, and/or payment; content control and administration; and human factors (e.g. user interfaces).
- support the operation of a plurality of clearinghouses, including, for example, both financial and user clearinghouse activities, such as

those performed by a client administrator in a large organization to assist in the organization's use of a VDE arrangement, including usage information analysis, and control of VDE activities by

5 individuals and groups of employees such as specifying budgets and the character of usage rights available under VDE for certain groups of and/or individual, client personnel, subject to control

10 information series to control information submitted by the client administrator. At a clearinghouse, one or more VDE installations may operate together with a trusted distributed database environment (which may include concurrent database processing

15 means). A financial clearinghouse normally receives at its location securely delivered content usage information, and user requests (such as requests for further credit, electronic currency, and/or higher credit limit). Reporting of usage information and

20 user requests can be used for supporting electronic currency, billing, payment and credit related activities, and/or for user profile analysis and/or broader market survey analysis and marketing (consolidated) list generation or other information derived, at least in part, from said usage

information. this information can be provided to content providers or other parties, through secure, authenticated encrypted communication to the VDE installation secure subsystems. Clearinghouse processing means would normally be connected to specialized I/O means, which may include high speed telecommunication switching means that may be used for secure communications between a clearinghouse and other VDE pathway participants.

5

10

- securely support electronic currency and credit usage control, storage, and communication at, and between, VDE installations. VDE further supports automated passing of electronic currency and/or credit information, including payment tokens (such as in the form of electronic currency or credit) or other payment information, through a pathway of payment, which said pathway may or may not be the same as a pathway for content usage information reporting. Such payment may be placed into a VDE container created automatically by a VDE installation in response to control information stipulating the "withdrawal" of credit or electronic currency from an electronic credit or currency

15

20

5 account based upon an amount owed resulting from
usage of VDE controlled electronic content and/or
appliances. Payment credit or currency may then be
automatically communicated in protected (at least in
part encrypted) form through telecommunication of
a VDE container to an appropriate party such as a
clearinghouse, provider of original property content
or appliance, or an agent for such provider (other
than a clearinghouse). Payment information may be
10 packaged in said VDE content container with, or
without, related content usage information, such as
metering information. An aspect of the present
invention further enables certain information
regarding currency use to be specified as
15 unavailable to certain, some, or all VDE parties
("conditionally" to fully anonymous currency) and/or
further can regulate certain content information,
such as currency and/or credit use related
information (and/or other electronic information
20 usage data) to be available only under certain strict
circumstances, such as a court order (which may
itself require authorization through the use of a
court controlled VDE installation that may be
required to securely access "conditionally"

anonymous information). Currency and credit information, under the preferred embodiment of the present invention, is treated as administrative content;

5

- support fingerprinting (also known as watermarking) for embedding in content such that when content protected under the present invention is released in clear form from a VDE object (displayed, printed, communicated, extracted, and/or saved), information representing the identification of the user and/or VDE installation responsible for transforming the content into clear form is embedded into the released content. Fingerprinting is useful in providing an ability to identify who extracted information in clear form a VDE container, or who made a copy of a VDE object or a portion of its contents. Since the identity of the user and/or other identifying information may be embedded in an obscure or generally concealed manner, in VDE container content and/or control information, potential copyright violators may be deterred from unauthorized extraction or copying. Fingerprinting normally is embedded into

10

15

20

unencrypted electronic content or control
information, though it can be embedded into
encrypted content and later placed in unencrypted
content in a secure VDE installation sub-system as
5 the encrypted content carrying the fingerprinting
information is decrypted. Electronic information,
such as the content of a VDE container, may be
fingerprinted as it leaves a network (such as
Internet) location bound for a receiving party. Such
10 repository information may be maintained in
unencrypted form prior to communication and be
encrypted as it leaves the repository. Fingerprinting
would preferably take place as the content leaves
the repository, but before the encryption step.
15 Encrypted repository content can be decrypted, for
example in a secure VDE sub-system, fingerprint
information can be inserted, and then the content
can be re-encrypted for transmission. Embedding
identification information of the intended recipient
20 user and/or VDE installation into content as it
leaves, for example, an Internet repository, would
provide important information that would identify
or assist in identifying any party that managed to
compromise the security of a VDE installation or the

5 delivered content. If a party produces an authorized clear form copy of VDE controlled content, including making unauthorized copies of an authorized clear form copy, fingerprint information would point back to that individual and/or his or her VDE installation. Such hidden information will act as a strong disincentive that should dissuade a substantial portion of potential content "pirates" from stealing other parties electronic information.

10 Fingerprint information identifying a receiving party and/or VDE installation can be embedded into a VDE object before, or during, decryption, replication, or communication of VDE content objects to receivers. Fingerprinting electronic

15 content before it is encrypted for transfer to a customer or other user provides information that can be very useful for identifying who received certain content which may have then been distributed or made available in unencrypted form.

20 This information would be useful in tracking who may have "broken" the security of a VDE installation and was illegally making certain electronic content available to others. Fingerprinting may provide additional, available

information such as time and/or date of the release
(for example extraction) of said content information.
Locations for inserting fingerprints may be specified
by VDE installation and/or content container control
5 information. This information may specify that
certain areas and/or precise locations within
properties should be used for fingerprinting, such as
one or more certain fields of information or
information types. Fingerprinting information may
10 be incorporated into a property by modifying in a
normally undetectable way color frequency and/or
the brightness of certain image pixels, by slightly
modifying certain audio signals as to frequency, by
modifying font character formation, etc.
15 Fingerprint information, itself, should be encrypted
so as to make it particularly difficult for tampered
fingerprints to be interpreted as valid. Variations in
fingerprint locations for different copies of the same
property; "false" fingerprint information; and
20 multiple copies of fingerprint information within a
specific property or other content which copies
employ different fingerprinting techniques such as
information distribution patterns, frequency and/or
brightness manipulation, and encryption related

techniques, are features of the present invention for increasing the difficulty of an unauthorized individual identifying fingerprint locations and erasing and/or modifying fingerprint information.

5

- provide smart object agents that can carry requests, data, and/or methods, including budgets, authorizations, credit or currency, and content. For example, smart objects may travel to and/or from remote information resource locations and fulfill requests for electronic information content. Smart objects can, for example, be transmitted to a remote location to perform a specified database search on behalf of a user or otherwise "intelligently" search remote one or more repositories of information for user desired information. After identifying desired information at one or more remote locations, by for example, performing one or more database searches, a smart object may return via communication to the user in the form of a secure "return object" containing retrieved information. A user may be charged for the remote retrieving of information, the returning of information to the user's VDE installation, and/or the use of such information. In

10

15

20

the latter case, a user may be charged only for the information in the return object that the user actually uses. Smart objects may have the means to request use of one or more services and/or resources.

5 Services include locating other services and/or resources such as information resources, language or format translation, processing, credit (or additional credit) authorization, etc. Resources include

10 reference databases, networks, high powered or specialized computing resources (the smart object may carry information to another computer to be efficiently processed and then return the information to the sending VDE installation),

15 remote object repositories, etc. Smart objects can make efficient use of remote resources (e.g. centralized databases, super computers, etc.) while providing a secure means for charging users based on information and/or resources actually used.

20

- support both "translations" of VDE electronic agreements elements into modern language printed agreement elements (such as English language agreements) and translations of electronic rights protection/transaction management modern

language agreement elements to electronic VDE
agreement elements. This feature requires
maintaining a library of textual language that
corresponds to VDE load modules and/or methods
5 and/or component assemblies. As VDE methods are
proposed and/or employed for VDE agreements, a
listing of textual terms and conditions can be
produced by a VDE user application which, in a
preferred embodiment, provides phrases, sentences
10 and/or paragraphs that have been stored and
correspond to said methods and/or assemblies. This
feature preferably employs artificial intelligence
capabilities to analyze and automatically determine,
and/or assist one or more users to determine, the
15 proper order and relationship between the library
elements corresponding to the chosen methods
and/or assemblies so as to compose some or all
portions of a legal or descriptive document. One or
more users, and/or preferably an attorney (if the
20 document a legal, binding agreement), would review
the generated document material upon completion
and employ such additional textual information
and/or editing as necessary to describe non
electronic transaction elements of the agreement

and make any other improvements that may be necessary. These features further support employing modern language tools that allow one or more users to make selections from choices and provide answers to questions and to produce a VDE electronic agreement from such a process. This process can be interactive and the VDE agreement formulation process may employ artificial intelligence expert system technology that learns from responses and, where appropriate and based at least in part on said responses, provides further choices and/or questions which "evolves" the desired VDE electronic agreement.

15 • support the use of multiple VDE secure subsystems in a single VDE installation. Various security and/or performance advantages may be realized by employing a distributed VDE design within a single VDE installation. For example, designing a hardware based VDE secure subsystem into an electronic appliance VDE display device, and designing said subsystem's integration with said display device so that it is as close as possible to the point of display, will increase the security for video

materials by making it materially more difficult to “steal” decrypted video information as it moves from outside to inside the video system. Ideally, for example, a VDE secure hardware module would be

5 in the same physical package as the actual display monitor, such as within the packaging of a video monitor or other display device, and such device would be designed, to the extent commercially practical, to be as tamper resistant as reasonable.

10 As another example, embedding a VDE hardware module into an I/O peripheral may have certain advantages from the standpoint of overall system throughput. If multiple VDE instances are employed within the same VDE installation, these

15 instances will ideally share resources to the extent practical, such as VDE instances storing certain control information and content and/or appliance usage information on the same mass storage device and in the same VDE management database.

20

- requiring reporting and payment compliance by employing exhaustion of budgets and time ageing of keys. For example, a VDE commercial arrangement and associated content control information may

involve a content provider's content and the use of clearinghouse credit for payment for end-user usage of said content. Control information regarding said arrangement may be delivered to a user's (of said
5 content) VDE installation and/or said financial clearinghouse's VDE installation. Said control information might require said clearinghouse to prepare and telecommunicate to said content provider both content usage based information in a
10 certain form, and content usage payment in the form of electronic credit (such credit might be "owned" by the provider after receipt and used in lieu of the availability or adequacy of electronic currency) and/or electronic currency. This delivery of
15 information and payment may employ trusted VDE installation secure subsystems to securely, and in some embodiments, automatically, provide in the manner specified by said control information, said usage information and payment content. Features
20 of the present invention help ensure that a requirement that a clearinghouse report such usage information and payment content will be observed. For example, if one participant to a VDE electronic agreement fails to observe such information

reporting and/or paying obligation, another participant can stop the delinquent party from successfully participating in VDE activities related to such agreement. For example, if required usage information and payment was not reported as specified by content control information, the “injured” party can fail to provide, through failing to securely communicate from his VDE installation secure subsystem, one or more pieces of secure information necessary for the continuance of one or more critical processes. For example, failure to report information and/or payment from a clearinghouse to a content provider (as well as any security failures or other disturbing irregularities) can result in the content provider not providing key and/or budget refresh information to the clearinghouse, which information can be necessary to authorize use of the clearinghouse’s credit for usage of the provider’s content and which the clearinghouse would communicate to end-user’s during a content usage reporting communication between the clearinghouse and end-user. As another example, a distributor that failed to make payments and/or report usage information to a

5 content provider might find that their budget for
creating permissions records to distribute the
content provider's content to users, and/or a security
budget limiting one or more other aspect of their use
of the provider's content, are not being refreshed by
the content provider, once exhausted or timed-out
(for example, at a predetermined date). In these and
other cases, the offended party might decide not to
refresh time ageing keys that had "aged out." Such
10 a use of time aged keys has a similar impact as
failing to refresh budgets or time-aged
authorizations.

- 15 • support smart card implementations of the present
invention in the form of portable electronic
appliances, including cards that can be employed as
secure credit, banking, and/or money cards. A
feature of the present invention is the use of
portable VDEs as transaction cards at retail and
20 other establishments, wherein such cards can "dock"
with an establishment terminal that has a VDE
secure sub-system and/or an online connection to a
VDE secure and/or otherwise secure and compatible
subsystem, such as a "trusted" financial

clearinghouse (e.g., VISA, Mastercard). The VDE
card and the terminal (and/or online connection) can
securely exchange information related to a
transaction, with credit and/or electronic currency
being transferred to a merchant and/or
clearinghouse and transaction information flowing
back to the card. Such a card can be used for
transaction activities of all sorts. A docking station,
such as a PCMCIA connector on an electronic
appliance, such as a personal computer, can receive
a consumer's VDE card at home. Such a
station/card combination can be used for on-line
transactions in the same manner as a VDE
installation that is permanently installed in such an
electronic appliance. The card can be used as an
"electronic wallet" and contain electronic currency as
well as credit provided by a clearinghouse. The card
can act as a convergence point for financial activities
of a consumer regarding many, if not all, merchant,
banking, and on-line financial transactions,
including supporting home banking activities. A
consumer can receive his paycheck and/or
investment earnings and/or "authentic" VDE content
container secured detailed information on such

receipts, through on-line connections. A user can
send digital currency to another party with a VDE
arrangement, including giving away such currency.
A VDE card can retain details of transactions in a
5 highly secure and database organized fashion so
that financially related information is both
consolidated and very easily retrieved and/or
analyzed. Because of the VDE security, including
use of effective encryption, authentication, digital
10 signaturing, and secure database structures, the
records contained within a VDE card arrangement
may be accepted as valid transaction records for
government and/or corporate recordkeeping
requirements. In some embodiments of the present
15 invention a VDE card may employ docking station
and/or electronic appliance storage means and/or
share other VDE arrangement means local to said
appliance and/or available across a network, to
augment the information storage capacity of the
20 VDE card, by for example, storing dated, and/or
archived, backup information. Taxes relating to
some or all of an individual's financial activities may
be automatically computed based on "authentic"
information securely stored and available to said

VDE card. Said information may be stored in said card, in said docking station, in an associated electronic appliance, and/or other device operatively attached thereto, and/or remotely, such as at a remote server site. A card's data, e.g. transaction history, can be backed up to an individual's personal computer or other electronic appliance and such an appliance may have an integrated VDE installation of its own. A current transaction, recent transactions (for redundancy), or all or other selected card data may be backed up to a remote backup repository, such a VDE compatible repository at a financial clearinghouse, during each or periodic docking for a financial transaction and/or information communication such as a user/merchant transaction. Backing up at least the current transaction during a connection with another party's VDE installation (for example a VDE installation that is also on a financial or general purpose electronic network), by posting transaction information to a remote clearinghouse and/or bank, can ensure that sufficient backup is conducted to enable complete reconstruction of VDE card internal information in the event of a card failure or loss.

- support certification processes that ensure authorized interoperability between various VDE installations so as to prevent VDE arrangements and/or installations that unacceptably deviate in specification protocols from other VDE arrangements and/or installations from interoperating in a manner that may introduce security (integrity and/or confidentiality of VDE secured information), process control, and/or software compatibility problems. Certification validates the identity of VDE installations and/or their components, as well as VDE users.

Certification data can also serve as information that contributes to determining the decommissioning or other change related to VDE sites.

- support the separation of fundamental transaction control processes through the use of event (triggered) based method control mechanisms. These event methods trigger one or more other VDE methods (which are available to a secure VDE sub-system) and are used to carry out VDE managed transaction related processing. These triggered methods include independently (separably) and

securely processable component billing management methods, budgeting management methods, metering management methods, and related auditing management processes. As a result of this feature of the present invention, independent triggering of metering, auditing, billing, and budgeting methods, the present invention is able to efficiently, concurrently support multiple financial currencies (e.g. dollars, marks, yen) and content related budgets, and/or billing increments as well as very flexible content distribution models.

- support, complete, modular separation of the control structures related to (1) content event triggering, (2) auditing, (3) budgeting (including specifying no right of use or unlimited right of use), (4) billing, and (5) user identity (VDE installation, client name, department, network, and/or user, etc.). The independence of these VDE control structures provides a flexible system which allows plural relationships between two or more of these structures, for example, the ability to associate a financial budget with different event trigger structures (that are put in place to enable

controlling content based on its logical portions).

Without such separation between these basic VDE capabilities, it would be more difficult to efficiently maintain separate metering, budgeting, identification, and/or billing activities which involve the same, differing (including overlapping), or entirely different, portions of content for metering, billing, budgeting, and user identification, for example, paying fees associated with usage of content, performing home banking, managing advertising services, etc. VDE modular separation of these basic capabilities supports the programming of plural, "arbitrary" relationships between one or differing content portions (and/or portion units) and budgeting, auditing, and/or billing control information. For example, under VDE, a budget limit of \$200 dollars or 300 German Marks a month may be enforced for decryption of a certain database and 2 U.S. Dollars or 3 German Marks may be charged for each record of said database decrypted (depending on user selected currency). Such usage can be metered while an additional audit for user profile purposes can be prepared recording the identity of each filed displayed. Additionally,

further metering can be conducted regarding the number of said database bytes that have been decrypted, and a related security budget may prevent the decrypting of more than 5% of the total bytes of said database per year. The user may also, under VDE (if allowed by senior control information), collect audit information reflecting usage of database fields by different individuals and client organization departments and ensure that differing rights of access and differing budgets limiting database usage can be applied to these client individuals and groups. Enabling content providers and users to practically employ such diverse sets of user identification, metering, budgeting, and billing control information results, in part, from the use of such independent control capabilities. As a result, VDE can support great configurability in creation of plural control models applied to the same electronic property and the same and/or plural control models applied to differing or entirely different content models (for example, home banking versus electronic shopping).

Methods, Other Control Information, and VDE Objects

VDE control information (e.g., methods) that collectively control use of VDE managed properties (database, document, individual commercial product), are either shipped with the content itself (for example, in a content container) and/or one or more portions of such control information is shipped to distributors and/or other users in separably deliverable “administrative objects.” A subset of the methods for a property may in part be delivered with each property while one or more other subsets of methods can be delivered separately to a user or otherwise made available for use (such as being available remotely by telecommunication means). Required methods (methods listed as required for property and/or appliance use) must be available as specified if VDE controlled content (such as intellectual property distributed within a VDE content container) is to be used. Methods that control content may apply to a plurality of VDE container objects, such as a class or other grouping of such objects. Methods may also be required by certain users or classes of users and/or VDE installations and/or classes of installations for such parties to use one or more specific, or classes of, objects.

A feature of VDE provided by the present invention is that certain one or more methods can be specified as required in order

for a VDE installation and/or user to be able to use certain
and/or all content. For example, a distributor of a certain type of
content might be allowed by "senior" participants (by content
creators, for example) to require a method which prohibits
5 end-users from electronically saving decrypted content, a
provider of credit for VDE transactions might require an audit
method that records the time of an electronic purchase, and/or a
user might require a method that summarizes usage information
for reporting to a clearinghouse (e.g. billing information) in a
10 way that does not convey confidential, personal information
regarding detailed usage behavior.

A further feature of VDE provided by the present
invention is that creators, distributors, and users of content can
15 select from among a set of predefined methods (if available) to
control container content usage and distribution functions and/or
they may have the right to provide new customized methods to
control at least certain usage functions (such "new" methods may
be required to be certified for trustedness and interoperability to
20 the VDE installation and/or for of a group of VDE applications).
As a result, VDE provides a very high degree of configurability
with respect to how the distribution and other usage of each
property or object (or one or more portions of objects or properties
as desired and/or applicable) will be controlled. Each VDE

participant in a VDE pathway of content control information
may set methods for some or all of the content in a VDE
container, so long as such control information does not conflict
with senior control information already in place with respect to:

5

- (1) certain or all VDE managed content,
- (2) certain one or more VDE users and/or groupings of
users,
- (3) certain one or more VDE nodes and/or groupings of
nodes, and/or
- (4) certain one or more VDE applications and/or
arrangements.

10

15

For example, a content creator's VDE control information
for certain content can take precedence over other submitted
VDE participant control information and, for example, if allowed
by senior control information, a content distributor's control
information may itself take precedence over a client
administrator's control information, which may take precedence
over an end-user's control information. A path of distribution
participant's ability to set such electronic content control

20

information can be limited to certain control information (for example, method mediating data such as pricing and/or sales dates) or it may be limited only to the extent that one or more of the participant's proposed control information conflicts with
5 control information set by senior control information submitted previously by participants in a chain of handling of the property, or managed in said participant's VDE secure subsystem.

VDE control information may, in part or in full, (a)
10 represent control information directly put in place by VDE content control information pathway participants, and/or (b) comprise control information put in place by such a participant on behalf of a party who does not directly handle electronic content (or electronic appliance) permissions records information
15 (for example control information inserted by a participant on behalf of a financial clearinghouse or government agency). Such control information methods (and/or load modules and/or mediating data and/or component assemblies) may also be put in place by either an electronic automated, or a semi-automated
20 and human assisted, control information (control set) negotiating process that assesses whether the use of one or more pieces of submitted control information will be integrated into and/or replace existing control information (and/or chooses between alternative control information based upon interaction with

in-place control information) and how such control information may be used.

5 Control information may be provided by a party who does not directly participate in the handling of electronic content (and/or appliance) and/or control information for such content (and/or appliance). Such control information may be provided in secure form using VDE installation secure sub-system managed communications (including, for example, authenticating the
10 deliverer of at least in part encrypted control information) between such not directly participating one or more parties' VDE installation secure subsystems, and a pathway of VDE content control information participant's VDE installation secure subsystem. This control information may relate to, for example,
15 the right to access credit supplied by a financial services provider, the enforcement of regulations or laws enacted by a government agency, or the requirements of a customer of VDE managed content usage information (reflecting usage of content by one or more parties other than such customer) relating to the
20 creation, handling and/or manner of reporting of usage information received by such customer. Such control information may, for example, enforce societal requirements such as laws related to electronic commerce.

VDE content control information may apply differently to different pathway of content and/or control information handling participants. Furthermore, permissions records rights may be added, altered, and/or removed by a VDE participant if they are allowed to take such action. Rights of VDE participants may be defined in relation to specific parties and/or categories of parties and/or other groups of parties in a chain of handling of content and/or content control information (e.g., permissions records). Modifications to control information that may be made by a given, eligible party or parties, may be limited in the number of modifications, and/or degree of modification, they may make.

At least one secure subsystem in electronic appliances of creators, distributors, auditors, clearinghouses, client administrators, and end-users (understanding that two or more of the above classifications may describe a single user) provides a "sufficiently" secure (for the intended applications) environment for:

1. Decrypting properties and control information;
2. Storing control and metering related information;
3. Managing communications;

4. Processing core control programs, along with associated data, that constitute control information for electronic content and/or appliance rights protection, including the enforcing of preferences and requirements of VDE participants.

5

10

15

20

Normally, most usage, audit, reporting, payment, and distribution control methods are themselves at least in part encrypted and are executed by the secure subsystem of a VDE installation. Thus, for example, billing and metering records can be securely generated and updated, and encryption and decryption keys are securely utilized, within a secure subsystem. Since VDE also employs secure (e.g. encrypted and authenticated) communications when passing information between the participant location (nodes) secure subsystems of a VDE arrangement, important components of a VDE electronic agreement can be reliably enforced with sufficient security (sufficiently trusted) for the intended commercial purposes. A VDE electronic agreement for a value chain can be composed, at least in part, of one or more subagreements between one or more subsets of the value chain participants. These subagreements are comprised of one or more electronic contract "compliance" elements (methods including associated parameter data) that ensure the protection of the rights of VDE participants.

The degree of trustedness of a VDE arrangement will be primarily based on whether hardware SPUs are employed at participant location secure subsystems and the effectiveness of the SPU hardware security architecture, software security techniques when an SPU is emulated in software, and the encryption algorithm(s) and keys that are employed for securing content, control information, communications, and access to VDE node (VDE installation) secure subsystems. Physical facility and user identity authentication security procedures may be used instead of hardware SPUs at certain nodes, such as at an established financial clearinghouse, where such procedures may provide sufficient security for trusted interoperability with a VDE arrangement employing hardware SPUs at user nodes.

The updating of property management files at each location of a VDE arrangement, to accommodate new or modified control information, is performed in the VDE secure subsystem and under the control of secure management file updating programs executed by the protected subsystem. Since all secure communications are at least in part encrypted and the processing inside the secure subsystem is concealed from outside observation and interference, the present invention ensures that content control information can be enforced. As a result, the creator and/or distributor and/or client administrator and/or

other contributor of secure control information for each property (for example, an end-user restricting the kind of audit information he or she will allow to be reported and/or a financial clearinghouse establishing certain criteria for use of its credit for payment for use of distributed content) can be confident that their contributed and accepted control information will be enforced (within the security limitations of a given VDE security implementation design). This control information can determine, for example:

10

(1) How and/or to whom electronic content can be provided, for example, how an electronic property can be distributed;

15

(2) How one or more objects and/or properties, or portions of an object or property, can be directly used, such as decrypted, displayed, printed, etc;

20

(3) How payment for usage of such content and/or content portions may or must be handled; and

(4) How audit information about usage information related to at least a portion of a property should be collected, reported, and/or used.

Seniority of contributed control information, including resolution of conflicts between content control information submitted by multiple parties, is normally established by:

- 5 (1) the sequence in which control information is put in place by various parties (in place control information normally takes precedence over subsequently submitted control information),

- 10 (2) the specifics of VDE content and/or appliance control information. For example, in-place control information can stipulate which subsequent one or more piece of control from one or more parties or class of parties will take precedence over control
- 15 information submitted by one or more yet different parties and/or classes of parties, and/or

- 20 (3) negotiation between control information sets from plural parties, which negotiation establishes what control information shall constitute the resulting control information set for a given piece of VDE managed content and/or VDE installation.

Electronic Agreements and Rights Protection

An important feature of VDE is that it can be used to assure the administration of, and adequacy of security and rights protection for, electronic agreements implemented through the use of the present invention. Such agreements may involve one or more of:

- (1) creators, publishers, and other distributors, of electronic information,
- (2) financial service (e.g. credit) providers,
- (3) users of (other than financial service providers) information arising from content usage such as content specific demographic information and user specific descriptive information. Such users may include market analysts, marketing list compilers for direct and directed marketing, and government agencies,
- (4) end users of content,
- (5) infrastructure service and device providers such as telecommunication companies and hardware

manufacturers (semiconductor and electronic
appliance and/or other computer system
manufacturers) who receive compensation based
upon the use of their services and/or devices, and

5

(6) certain parties described by electronic information.

VDE supports commercially secure "extended" value chain
electronic agreements. VDE can be configured to support the
various underlying agreements between parties that comprise
this extended agreement. These agreements can define
important electronic commerce considerations including:

10

(1) security,

15

(2) content use control, including electronic distribution,

20

(3) privacy (regarding, for example, information
concerning parties described by medical, credit, tax,
personal, and/or of other forms of confidential
information),

(4) management of financial processes, and

- (5) pathways of handling for electronic content, content and/or appliance control information, electronic content and/or appliance usage information and payment and/or credit.

5

VDE agreements may define the electronic commerce relationship of two or more parties of a value chain, but such agreements may, at times, not directly obligate or otherwise directly involve other VDE value chain participants. For example, an electronic agreement between a content creator and a distributor may establish both the price to the distributor for a creator's content (such as for a property distributed in a VDE container object) and the number of copies of this object that this distributor may distribute to end-users over a given period of time. In a second agreement, a value chain end-user may be involved in a three party agreement in which the end-user agrees to certain requirements for using the distributed product such as accepting distributor charges for content use and agreeing to observe the copyright rights of the creator. A third agreement might exist between the distributor and a financial clearinghouse that allows the distributor to employ the clearinghouse's credit for payment for the product if the end-user has a separate (fourth) agreement directly with the clearinghouse extending credit to the end-user. A fifth, evolving

10

15

20

agreement may develop between all value chain participants as content control information passes along its chain of handling. This evolving agreement can establish the rights of all parties to content usage information, including, for example, the nature of information to be received by each party and the pathway of handling of content usage information and related procedures. A sixth agreement in this example, may involve all parties to the agreement and establishes certain general assumptions, such as security techniques and degree of trustedness (for example, commercial integrity of the system may require each VDE installation secure subsystem to electronically warrant that their VDE node meets certain interoperability requirements). In the above example, these six agreements could comprise agreements of an extended agreement for this commercial value chain instance.

VDE agreements support evolving ("living") electronic agreement arrangements that can be modified by current and/or new participants through very simple to sophisticated "negotiations" between newly proposed content control information interacting with control information already in place and/or by negotiation between concurrently proposed content control information submitted by a plurality of parties. A given model may be asynchronously and progressively modified over

time in accordance with existing senior rules and such modification may be applied to all, to classes of, and/or to specific content, and/or to classes and/or specific users and/or user nodes. A given piece of content may be subject to different control

5 information at different times or places of handling, depending on the evolution of its content control information (and/or on differing, applicable VDE installation content control information). The evolution of control information can occur during the passing along of one or more VDE control information

10 containing objects, that is control information may be modified at one or more points along a chain of control information handling, so long as such modification is allowed. As a result, VDE managed content may have different control information applied at both different "locations" in a chain of content handling and at

15 similar locations in differing chains of the handling of such content. Such different application of control information may also result from content control information specifying that a certain party or group of parties shall be subject to content control information that differs from another party or group of

20 parties. For example, content control information for a given piece of content may be stipulated as senior information and therefore not changeable, might be put in place by a content creator and might stipulate that national distributors of a given piece of their content may be permitted to make 100,000 copies

per calendar quarter, so long as such copies are provided to boni
fide end-users, but may pass only a single copy of such content to
a local retailers and the control information limits such a retailer
to making no more than 1,000 copies per month for retail sales to
5 end-users. In addition, for example, an end-user of such content
might be limited by the same content control information to
making three copies of such content, one for each of three
different computers he or she uses (one desktop computer at
work, one for a desktop computer at home, and one for a portable
10 computer).

Electronic agreements supported by the preferred
embodiment of the present invention can vary from very simple
to very elaborate. They can support widely diverse information
15 management models that provide for electronic information
security, usage administration, and communication and may
support:

- 20 (a) secure electronic distribution of information, for
example commercial literary properties,
- (b) secure electronic information usage monitoring and
reporting,

- 5
- (c) secure financial transaction capabilities related to both electronic information and/or appliance usage and other electronic credit and/or currency usage and administration capabilities,
- (d) privacy protection for usage information a user does not wish to release, and
- 10 (e) "living" electronic information content dissemination models that flexibly accommodate:
- (1) a breadth of participants,
- 15 (2) one or more pathways (chains) for: the handling of content, content and/or appliance control information, reporting of content and/or appliance usage related information, and/or payment,
- 20 (3) supporting an evolution of terms and conditions incorporated into content control information, including use of electronic negotiation capabilities,

- (4) support the combination of multiple pieces of content to form new content aggregations, and
- (5) multiple concurrent models.

5

Secure Processing Units

An important part of VDE provided by the present invention is the core secure transaction control arrangement, herein called an SPU (or SPUs), that typically must be present in each user's computer, other electronic appliance, or network.

SPUs provide a trusted environment for generating decryption keys, encrypting and decrypting information, managing the secure communication of keys and other information between electronic appliances (i.e. between VDE installations and/or between plural VDE instances within a single VDE installation), securely accumulating and managing audit trail, reporting, and budget information in secure and/or non-secure non-volatile memory, maintaining a secure database of control information management instructions, and providing a secure environment for performing certain other control and administrative functions.

A hardware SPU (rather than a software emulation) within a VDE node is necessary if a highly trusted environment

for performing certain VDE activities is required. Such a trusted environment may be created through the use of certain control software, one or more tamper resistant hardware modules such as a semiconductor or semiconductor chipset (including, for example, a tamper resistant hardware electronic appliance peripheral device), for use within, and/or operatively connected to, an electronic appliance. With the present invention, the trustedness of a hardware SPU can be enhanced by enclosing some or all of its hardware elements within tamper resistant packaging and/or by employing other tamper resisting techniques (e.g. microfusing and/or thin wire detection techniques). A trusted environment of the present invention implemented, in part, through the use of tamper resistant semiconductor design, contains control logic, such as a microprocessor, that securely executes VDE processes.

A VDE node's hardware SPU is a core component of a VDE secure subsystem and may employ some or all of an electronic appliance's primary control logic, such as a microcontroller, microcomputer or other CPU arrangement. This primary control logic may be otherwise employed for non VDE purposes such as the control of some or all of an electronic appliance's non-VDE functions. When operating in a hardware SPU mode, said primary control logic must be sufficiently secure so as to protect

and conceal important VDE processes. For example, a hardware SPU may employ a host electronic appliance microcomputer operating in protected mode while performing VDE related activities, thus allowing portions of VDE processes to execute

5 with a certain degree of security. This alternate embodiment is in contrast to the preferred embodiment wherein a trusted environment is created using a combination of one or more tamper resistant semiconductors that are not part of said primary control logic. In either embodiment, certain control

10 information (software and parameter data) must be securely maintained within the SPU, and further control information can be stored externally and securely (e.g. in encrypted and tagged form) and loaded into said hardware SPU when needed. In many cases, and in particular with microcomputers, the

15 preferred embodiment approach of employing special purpose secure hardware for executing said VDE processes, rather than using said primary control logic, may be more secure and efficient. The level of security and tamper resistance required for trusted SPU hardware processes depends on the commercial

20 requirements of particular markets or market niches, and may vary widely.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features and advantages provided by the present invention(s) may be better and more completely understood by referring to the following detailed description of presently preferred example embodiments in connection with the drawings, of which:

FIGURE 1 illustrates an example of a "Virtual Distribution Environment" provided in accordance with a preferred example/embodiment of this invention;

FIGURE 1A is a more detailed illustration of an example of the "Information Utility" shown in FIGURE 1;

FIGURE 2 illustrates an example of a chain of handling and control;

FIGURE 2A illustrates one example of how rules and control information may persist from one participant to another in the Figure 2 chain of handling and control;

FIGURE 3 shows one example of different control information that may be provided;

FIGURE 4 illustrates examples of some different types of rules and/or control information;

FIGURES 5A and 5B show an example of an "object";

5

FIGURE 6 shows an example of a Secure Processing Unit ("SPU");

FIGURE 7 shows an example of an electronic appliance;

10

FIGURE 8 is a more detailed block diagram of an example of the electronic appliance shown in FIGURE 7;

FIGURE 9 is a detailed view of an example of the Secure Processing Unit (SPU) shown in FIGURES 6 and 8;

15

Figure 9A shows an example combined secure processing unit and control processing unit;

20

Figure 9B shows an example secure processing unit integrated with a standard CPU;

FIGURE 10 shows an example of a "Rights Operating System" ("ROS") architecture provided by the Virtual Distribution Environment;

5 FIGURES 11A-11C show examples of functional relationship(s) between applications and the Rights Operating System;

10 FIGURES 11D-11J show examples of "components" and "component assemblies";

FIGURE 12 is a more detailed diagram of an example of the Rights Operating System shown in FIGURE 10;

15 FIGURE 12A shows an example of how "objects" can be created;

20 FIGURE 13 is a detailed block diagram of an example the software architecture for a "protected processing environment" shown in FIGURE 12;

FIGURES 14A-14C are examples of SPU memory maps provided by the protected processing environment shown in FIGURE 13;

FIGURE 15 illustrates an example of how the channel services manager and load module execution manager of FIGURE 13 can support a channel;

5 FIGURE 15A is an example of a channel header and channel detail records shown in FIGURE 15;

 FIGURE 15B is a flowchart of an example of program control steps that may be performed by the FIGURE 13 protected
10 processing environment to create a channel;

 FIGURE 16 is a block diagram of an example of a secure data base structure;

15 FIGURE 17 is an illustration of an example of a logical object structure;

 FIGURE 18 shows an example of a stationary object structure;

20 FIGURE 19 shows an example of a traveling object structure;

FIGURE 20 shows an example of a content object
structure;

5 FIGURE 21 shows an example of an administrative object
structure;

FIGURE 22 shows an example of a method core structure;

10 FIGURE 23 shows an example of a load module structure;

FIGURE 24 shows an example of a User Data Element
(UDE) and/or Method Data Element (MDE) structure;

15 FIGURES 25A-25C show examples of "map meters";

FIGURE 26 shows an example of a permissions record
(PERC) structure;

20 FIGURES 26A and 26B together show a more detailed
example of a permissions record structure;

FIGURE 27 shows an example of a shipping table
structure;

FIGURE 28 shows an example of a receiving table structure;

5 FIGURE 29 shows an example of an administrative event log structure;

FIGURE 30 shows an example inter-relationship between and use of the object registration table, subject table and user rights table shown in the FIGURE 16 secure database;

10

FIGURE 31 is a more detailed example of an object registration table shown in FIGURE 16;

15 FIGURE 32 is a more detailed example of subject table shown in FIGURE 16;

FIGURE 33 is a more detailed example of a user rights table shown in FIGURE 16;

20 FIGURE 34 shows a specific example of how a site record table and group record table may track portions of the secure database shown in FIGURE 16;

FIGURE 34A is an example of a FIGURE 34 site record table structure;

5 FIGURE 34B is an example of a FIGURE 34 group record table structure;

FIGURE 35 shows an example of a process for updating the secure database;

10 FIGURE 36 shows an example of how new elements may be inserted into the FIGURE 16 secure data base;

FIGURE 37 shows an example of how an element of the secure database may be accessed;

15

FIGURE 38 is a flowchart example of how to protect a secure database element;

20 FIGURE 39 is a flowchart example of how to back up a secure database;

FIGURE 40 is a flowchart example of how to recover a secure database from a backup;

FIGURES 41A-41D are a set of examples showing how a “chain of handling and control” may be enabled using “reciprocal methods”;

5 FIGURES 42A-42D show an example of a “reciprocal”
BUDGET method;

FIGURES 43A-43D show an example of a “reciprocal”
REGISTER method;

10

FIGURES 44A-44C show an example of a “reciprocal”
AUDIT method;

15 FIGURES 45-48 show examples of several methods being
used together to control release of content or other information;

FIGURES 49, 49A-49F show an example OPEN method;

20 FIGURES 50, 50A-50F show an example of a READ
method;

FIGURES 51, 51A-51F show an example of a WRITE
method;

FIGURE 52 shows an example of a CLOSE method;

FIGURES 53A-53B show an example of an EVENT
method;

5

FIGURE 53C shows an example of a BILLING method;

FIGURE 54 shows an example of an ACCESS method;

10

FIGURES 55A-55B show examples of DECRYPT and
ENCRYPT methods;

FIGURE 56 shows an example of a CONTENT method;

15

FIGURES 57A and 57B show examples of EXTRACT and
EMBED methods;

FIGURE 58A shows an example of an OBSCURE method;

20

FIGURES 58B, 58C show examples of a FINGERPRINT
method;

FIGURE 59 shows an example of a DESTROY method;

FIGURE 60 shows an example of a PANIC method;

FIGURE 61 shows an example of a METER method;

5 FIGURE 62 shows an example of a key "convolution"
process;

FIGURE 63 shows an example of how different keys may
be generated using a key convolution process to determine a
10 "true" key;

FIGURES 64 and 65 show an example of how protected
processing environment keys may be initialized;

15 FIGURES 66 and 67 show example processes for
decrypting information contained within stationary and
traveling objects, respectively;

Figures 67A and 67B show example techniques for
20 cracking a software-based protected processing environment;

FIGURE 68 shows an example of how a protected
processing environment may be initialized;

FIGURE 69 shows an example of how firmware may be downloaded into a protected processing environment;

5 Figure 69A shows an example technique for distributing protected processing environment software;

Figure 69B-69C show an example installation routine for installing a software-based protected processing environment;

10 Figure 69D shows example techniques for embedding cryptographic keys at random locations within structure-based protected processing environment operational materials;

15 Figure 69E shows example locations for PPE operational materials random modifications and/or digital fingerprints;

Figure 69F shows an example customized static storage layout for PPE operational materials;

20 Figure 69G shows example electronic appliance signature locations;

Figure 69H shows example sequence dependent and independent processes;

Figures 69I and 69J show example static code and data storage organizations;

5 Figures 69K-69L together show example steps for providing dynamic protection mechanisms;

Figure 69M shows an example initialization time check routine;

10 Figure 69N shows an example time check routine;

Figure 69O shows example time check data structures;

15 FIGURE 70 shows an example of multiple VDE electronic appliances connected together with a network or other communications means;

20 Figure 70A shows how content may be prepared for printing and encrypted inside a PPE, then decrypted inside a printer;

Figure 70B shows how characters may be selected from slightly different fonts in order to place an electronic fingerprint or watermark into printed output;

Figure 70C shows how characters in a font may be permuted to render a printed page unusable without the corresponding scrambled font;

5 FIGURE 71 shows an example of a portable VDE electronic appliance;

 FIGURES 72A-72D show examples of "pop-up" displays that may be generated by the user notification and exception
10 interface;

 FIGURE 73 shows an example of a "smart object";

 FIGURE 74 shows an example of a process using "smart
15 objects";

 FIGURES 75A-75D show examples of data structures used for electronic negotiation;

20 FIGURES 75E-75F show example structures relating to an electronic agreement;

 FIGURES 76A-76B show examples of electronic negotiation processes;

FIGURE 77 shows a further example of a chain of handling and control;

5

FIGURE 78 shows an example of a VDE "repository";

FIGURES 79-83 show an example illustrating a chain of handling and control to evolve and transform VDE managed content and control information;

10

FIGURE 84 shows a further example of a chain of handling and control involving several categories of VDE participants;

15

FIGURE 85 shows a further example of a chain of distribution and handling within an organization;

Figures 86 and 86A show a further example of a chain of handling and control; and

20

Figure 87 shows an example of a virtual silicon container model.

MORE DETAILED DESCRIPTION

Figures 1-7 and the discussion below provides an overview of some aspects of features provided by this invention. Following this overview is a more technical "detail description" of example embodiments in accordance with the invention.

5

Overview

Figure 1 shows a "Virtual Distribution Environment" ("VDE") 100 that may be provided in accordance with this invention. In Figure 1, an information utility 200 connects to communications means 202 such as telephone or cable TV lines for example. Telephone or cable TV lines 202 may be part of an "electronic highway" that carries electronic information from place to place. Lines 202 connect information utility 200 to other people such as for example a consumer 208, an office 210, a video production studio 204, and a publishing house 214. Each of the people connected to information utility 200 may be called a "VDE participant" because they can participate in transactions occurring within the virtual distribution environment 100.

15
20

Almost any sort of transaction you can think of can be supported by virtual distribution environment 100. A few of many examples of transactions that can be supported by virtual distribution environment 100 include:

- home banking and electronic payments;
- electronic legal contracts;
- distribution of “content” such as electronic printed matter, video, audio, images and computer programs; and
- 5 • secure communication of private information such as medical records and financial information.

Virtual distribution environment 100 is “virtual” because it does not require many of the physical “things” that used to be necessary to protect rights, ensure reliable and predictable
10 distribution, and ensure proper compensation to content creators and distributors. For example, in the past, information was distributed on records or disks that were difficult to copy. In the past, private or secret content was distributed in sealed
15 envelopes or locked briefcases delivered by courier. To ensure appropriate compensation, consumers received goods and services only after they handed cash over to a seller. Although information utility 200 may deliver information by transferring physical “things” such as electronic storage media, the virtual
20 distribution environment 100 facilitates a completely electronic “chain of handling and control.”

VDE Flexibility Supports Transactions

Information utility 200 flexibly supports many different kinds of information transactions. Different VDE participants may define and/or participate in different parts of a transaction. Information utility 200 may assist with delivering information about a transaction, or it may be one of the transaction participants.

For example, the video production studio 204 in the upper right-hand corner of Figure 1 may create video/television programs. Video production studio 204 may send these programs over lines 202, or may use other paths such as satellite link 205 and CD ROM delivery service 216. Video production studio 204 can send the programs directly to consumers 206, 208, 210, or it can send the programs to information utility 200 which may store and later send them to the consumers, for example. Consumers 206, 208, 210 are each capable of receiving and using the programs created by video production studio 204—assuming, that is, that the video production studio or information utility 200 has arranged for these consumers to have appropriate “rules and controls” (control information) that give the consumers rights to use the programs.

Even if a consumer has a copy of a video program, she cannot watch or copy the program unless she has “rules and

controls” that authorize use of the program. She can use the program only as permitted by the “rules and controls.”

5 For example, video production studio 204 might release a half-hour exercise video in the hope that as many viewers as possible will view it. The video production studio 204 wishes to receive \$2.00 per viewing. Video production studio 204 may, through information utility 200, make the exercise video available in “protected” form to all consumers 206, 208, 210.

10 Video production studio 204 may also provide “rules and controls” for the video. These “rules and controls” may specify for example:

15 (1) any consumer who has good credit of at least \$2.00 based on a credit account with independent financial provider 212 (such as Mastercard or VISA) may watch the video,

20 (2) virtual distribution environment 100 will “meter” each time a consumer watches the video, and report usage to video production studio 204 from time to time, and

(3) financial provider 212 may electronically collect payment (\$2.00) from the credit account of each consumer

who watches the video, and transfer these payments to the video production studio 204.

Information utility 200 allows even a small video
5 production studio to market videos to consumers and receive compensation for its efforts. Moreover, the videos can, with appropriate payment to the video production studio, be made available to other video publishers who may add value and/or act as repackagers or redistributors.

10

Figure 1 also shows a publishing house 214. Publishing house 214 may act as a distributor for an author 206. The publishing house 214 may distribute rights to use "content" (such as computer software, electronic newspapers, the video produced
15 by publishing house 214, audio, or any other data) to consumers such as office 210. The use rights may be defined by "rules and controls" distributed by publishing house 216. Publishing house 216 may distribute these "rules and controls" with the content, but this is not necessary. Because the content can be used only
20 by consumers that have the appropriate "rules and controls," content and its associated "rules and controls" may be distributed at different times, in different ways, by different VDE participants. The ability of VDE to securely distribute and

enforce "rules and controls" separately from the content they apply to provides great advantages.

5 Use rights distributed by publishing house 214 may, for example, permit office 210 to make and distribute copies of the content to its employees. Office 210 may act as a redistributor by extending a "chain of handling and control" to its employees. The office 210 may add or modify "rules and controls" (consistent with the "rules and controls" it receives from publishing house 10 214) to provide office-internal control information and mechanisms. For example, office 210 may set a maximum usage budget for each individual user and/or group within the office, or it may permit only specified employees and/or groups to access certain information.

15

Figure 1 also shows an information delivery service 216 delivering electronic storage media such as "CD ROM" disks to consumers 206. Even though the electronic storage media themselves are not delivered electronically by information utility 200 over lines 202, they are still part of the virtual distribution environment 100. The electronic storage media may be used to 20 distribute content, "rules and controls," or other information.

Example of What's Inside Information Utility 200

5 "Information utility" 200 in Figure 1 can be a collection of participants that may act as distributors, financial clearinghouses, and administrators. Figure 1A shows an example of what may be inside one example of information utility 200. Information utility participants 200a-200g could each be an independent organization/business. There can be any number of each of participants 200a-200g. In this example, electronic "switch" 200a connects internal parts of information utility 200 to each other and to outside participants, and may also connect outside participants to one another.

10

Information utility 200 may include a "transaction processor" 200b that processes transactions (to transfer electronic funds, for example) based on requests from participants and/or report receiver 200e. It may also include a "usage analyst" 200c that analyzes reported usage information. A "report creator" 200d may create reports based on usage for example, and may provide these reports to outside participants and/or to participants within information utility 200. A "report receiver" 200e may receive reports such as usage reports from content users. A "permissioning agent" 200f may distribute "rules and controls" granting usage or distribution permissions based on a profile of a consumer's credit worthiness, for example.

15

20

An administrator 200h may provide information that keeps the virtual distribution environment 100 operating properly. A content and message storage 200g may store information for use by participants within or outside of information utility 200.

5

Example of Distributing Content* Using A Chain of Handling and Control*

As explained above, virtual distribution environment 100 can be used to manage almost any sort of transaction. One type of important transaction that virtual distribution environment 100 may be used to manage is the distribution or communication of "content" or other important information. Figure 2 more abstractly shows a "model" of how the Figure 1 virtual distribution environment 100 may be used to provide a "chain of handling and control" for distributing content. Each of the blocks in Figure 2 may correspond to one or more of the VDE participants shown in Figure 1.

20

In the Figure 2 example, a VDE content creator 102 creates "content." The content creator 102 may also specify "rules and controls" for distributing the content. These distribution-related "rules and controls" can specify who has permission to distribute the rights to use content, and how many users are allowed to use the content.

25

Arrow 104 shows the content creator 102 sending the
“rules and controls” associated with the content to a VDE rights
distributor 106 (“distributor”) over an electronic highway 108 (or
by some other path such as an optical disk sent by a delivery
5 service such as U. S. mail). The content can be distributed over
the same or different path used to send the “rules and controls.”
The distributor 106 generates her own “rules and controls” that
relate to usage of the content. The usage-related “rules and
controls” may, for example, specify what a user can and can’t do
10 with the content and how much it costs to use the content. These
usage-related “rules and controls” must be consistent with the
“rules and controls” specified by content creator 102.

Arrow 110 shows the distributor 106 distributing rights to
15 use the content by sending the content’s “rules and controls” to a
content user 112 such as a consumer. The content user 112 uses
the content in accordance with the usage-related “rules and
controls.”

20 In this Figure 2 example, information relating to content
use is, as shown by arrow 114, reported to a financial
clearinghouse 116. Based on this “reporting,” the financial
clearinghouse 116 may generate a bill and send it to the content
user 112 over a “reports and payments” network 118. Arrow 120

shows the content user 112 providing payments for content usage to the financial clearinghouse 116. Based on the reports and payments it receives, the financial clearinghouse 116 may provide reports and/or payments to the distributor 106. The distributor 106 may, as shown by arrow 122, provide reports and/or payments to the content creator 102. The clearinghouse 116 may provide reports and payments directly to the creator 102. Reporting and/or payments may be done differently. For example, clearinghouse 116 may directly or through an agent, provide reports and/or payments to each of VDE content creators 102, and rights distributor 106, as well as reports to content user 112.

The distributor 106 and the content creator 102 may be the same person, or they may be different people. For example, a musical performing group may act as both content creator 102 and distributor 106 by creating and distributing its own musical recordings. As another example, a publishing house may act as a distributor 106 to distribute rights to use works created by an author content creator 102. Content creators 102 may use a distributor 106 to efficiently manage the financial end of content distribution.

The "financial clearinghouse" 116 shown in Figure 2 may also be a "VDE administrator." Financial clearinghouse 116 in its VDE administrator role sends "administrative" information to the VDE participants. This administrative information helps to keep the virtual distribution environment 100 operating properly. The "VDE administrator" and financial clearinghouse roles may be performed by different people or companies, and there can be more than one of each.

10 **More about Rules and Controls"**

The virtual distribution environment 100 prevents use of protected information except as permitted by the "rules and controls" (control information). For example, the "rules and controls" shown in Figure 2 may grant specific individuals or classes of content users 112 "permission" to use certain content. They may specify what kinds of content usage are permitted, and what kinds are not. They may specify how content usage is to be paid for and how much it costs. As another example, "rules and controls" may require content usage information to be reported back to the distributor 106 and/or content creator 102.

Every VDE participant in "chain of handling and control" is normally subject to "rules and controls." "Rules and controls" define the respective rights and obligations of each of the various

VDE participants. "Rules and controls" provide information and mechanisms that may establish interdependencies and relationships between the participants. "Rules and controls" are flexible, and permit "virtual distribution environment" 100 to support most "traditional" business transactions. For example:

- "Rules and controls" may specify which financial clearinghouse(s) 116 may process payments,
- "Rules and controls" may specify which participant(s) receive what kind of usage report, and
- "Rules and controls" may specify that certain information is revealed to certain participants, and that other information is kept secret from them.

"Rules and controls" may self limit if and how they may be changed. Often, "rules and controls" specified by one VDE participant cannot be changed by another VDE participant. For example, a content user 112 generally can't change "rules and controls" specified by a distributor 106 that require the user to pay for content usage at a certain rate. "Rules and controls" may "persist" as they pass through a "chain of handling and control," and may be "inherited" as they are passed down from one VDE participant to the next.

Depending upon their needs, VDE participants can specify that their "rules and controls" can be changed under conditions specified by the same or other "rules and controls." For example, "rules and controls" specified by the content creator 102 may
5 permit the distributor 106 to "mark up" the usage price just as retail stores "mark up" the wholesale price of goods. Figure 2A shows an example in which certain "rules and controls" persist unchanged from content creator 102 to content user 112; other "rules and controls" are modified or deleted by distributor 106;
10 and still other "rules and controls" are added by the distributor.

"Rules and controls" can be used to protect the content user's privacy by limiting the information that is reported to other VDE participants. As one example, "rules and controls"
15 can cause content usage information to be reported anonymously without revealing content user identity, or it can reveal only certain information to certain participants (for example, information derived from usage) with appropriate permission, if required. This ability to securely control what information is
20 revealed and what VDE participant(s) it is revealed to allows the privacy rights of all VDE participants to be protected.

Rules and Contents" Can Be Separately Delivered

As mentioned above, virtual distribution environment 100 "associates" content with corresponding "rules and controls," and prevents the content from being used or accessed unless a set of
5 corresponding "rules and controls" is available. The distributor 106 doesn't need to deliver content to control the content's distribution. The preferred embodiment can securely protect content by protecting corresponding, usage enabling "rules and controls" against unauthorized distribution and use.

10

In some examples, "rules and controls" may travel with the content they apply to. Virtual distribution environment 100 also allows "rules and controls" to be delivered separately from content. Since no one can use or access protected content
15 without "permission" from corresponding "rules and controls," the distributor 106 can control use of content that has already been (or will in the future be) delivered. "Rules and controls" may be delivered over a path different from the one used for content delivery. "Rules and controls" may also be delivered at
20 some other time. The content creator 102 might deliver content to content user 112 over the electronic highway 108, or could make the content available to anyone on the highway. Content may be used at the time it is delivered, or it may be stored for later use or reuse.

The virtual distribution environment 100 also allows payment and reporting means to be delivered separately. For example, the content user 112 may have a virtual "credit card" that extends credit (up to a certain limit) to pay for usage of any content. A "credit transaction" can take place at the user's site without requiring any "online" connection or further authorization. This invention can be used to help securely protect the virtual "credit card" against unauthorized use.

10 **Rules and Contents" Define Processes**

Figure 3 shows an example of an overall process based on "rules and controls." It includes an "events" process 402, a meter process 404, a billing process 406, and a budget process 408. Not all of the processes shown in Figure 3 will be used for every set of "rules and controls."

The "events process" 402 detects things that happen ("events") and determines which of those "events" need action by the other "processes." The "events" may include, for example, a request to use content or generate a usage permission. Some events may need additional processing, and others may not. Whether an "event" needs more processing depends on the "rules and controls" corresponding to the content. For example, a user who lacks permission will not have her request satisfied ("No

Go"). As another example, each user request to turn to a new page of an electronic book may be satisfied ("Go"), but it may not be necessary to meter, bill or budget those requests. A user who has purchased a copy of a novel may be permitted to open and

5 read the novel as many times as she wants to without any further metering, billing or budgeting. In this simple example, the "event process" 402 may request metering, billing and/or budgeting processes the first time the user asks to open the protected novel (so the purchase price can be charged to the

10 user), and treat all later requests to open the same novel as "insignificant events." Other content (for example, searching an electronic telephone directory) may require the user to pay a fee for each access.

15 "Meter" process 404 keeps track of events, and may report usage to distributor 106 and/or other appropriate VDE participant(s). Figure 4 shows that process 404 can be based on a number of different factors such as:

- 20
- (a) type of usage to charge for,
 - (b) what kind of unit to base charges on,
 - (c) how much to charge per unit,

(d) when to report, and

(e) how to pay.

These factors may be specified by the "rules and controls" that control the meter process.

5

Billing process 406 determines how much to charge for events. It records and reports payment information.

10 Budget process 408 limits how much content usage is permitted. For example, budget process 408 may limit the number of times content may be accessed or copied, or it may limit the number of pages or other amount of content that can be used based on, for example, the number of dollars available in a credit account. Budget process 408 records and reports financial
15 and other transaction information associated with such limits.

Content may be supplied to the user once these processes have been successfully performed.

20 **Containers and Objects***

Figure 5A shows how the virtual distribution environment 100, in a preferred embodiment, may package information elements (content) into a "container" 302 so the information can't be accessed except as provided by its "rules and controls."

Normally, the container 302 is electronic rather than physical. Electronic container 302 in one example comprises "digital" information having a well defined structure. Container 302 and its contents can be called an "object 300."

5

The Figure 5A example shows items "within" and enclosed by container 302. However, container 302 may "contain" items without those items actually being stored within the container. For example, the container 302 may reference items that are available elsewhere such as in other containers at remote sites. Container 302 may reference items available at different times or only during limited times. Some items may be too large to store within container 302. Items may, for example, be delivered to the user in the form of a "live feed" of video at a certain time. Even then, the container 302 "contains" the live feed (by reference) in this example.

10

15

Container 302 may contain information content 304 in electronic (such as "digital") form. Information content 304 could be the text of a novel, a picture, sound such as a musical performance or a reading, a movie or other video, computer software, or just about any other kind of electronic information you can think of. Other types of "objects" 300 (such as

20

“administrative objects”) may contain “administrative” or other information instead of or in addition to information content 304.

5 In the Figure 5A example, container 302 may also contain “rules and controls” in the form of:

- (a) a “permissions record” 808;
- (b) “budgets” 308; and
- (c) “other methods” 1000.

10 Figure 5B gives some additional detail about permissions record 808, budgets 308 and other methods 1000. The “permissions record” 808 specifies the rights associated with the object 300 such as, for example, who can open the container 302, who can use the object’s contents, who can distribute the object, and what other control mechanisms must be active. For
15 example, permissions record 808 may specify a user’s rights to use, distribute and/or administer the container 302 and its content. Permissions record 808 may also specify requirements to be applied by the budgets 308 and “other methods” 1000.
20 Permissions record 808 may also contain security related information such as scrambling and descrambling “keys.”

“Budgets” 308 shown in Figure 5B are a special type of “method” 1000 that may specify, among other things, limitations

on usage of information content 304, and how usage will be paid
for. Budgets 308 can specify, for example, how much of the total
information content 304 can be used and/or copied. The methods
310 may prevent use of more than the amount specified by a
5 specific budget.

“Other methods” 1000 define basic operations used by
“rules and controls.” Such “methods” 1000 may include, for
example, how usage is to be “metered,” if and how content 304
10 and other information is to be scrambled and descrambled, and
other processes associated with handling and controlling
information content 304. For example, methods 1000 may record
the identity of anyone who opens the electronic container 302,
and can also control how information content is to be charged
15 based on “metering.” Methods 1000 may apply to one or several
different information contents 304 and associated containers
302, as well as to all or specific portions of information content
304.

20 **Secure Processing Unit (SPU)**

The “VDE participants” may each have an “electronic
appliance.” The appliance may be or contain a computer. The
appliances may communicate over the electronic highway 108.
Figure 6 shows a secure processing unit (“SPU”) 500 portion of

the "electronic appliance" used in this example by each VDE participant. SPU 500 processes information in a secure processing environment 503, and stores important information securely. SPU 500 may be emulated by software operating in a host electronic appliance.

SPU 500 is enclosed within and protected by a "tamper resistant security barrier" 502. Security barrier 502 separates the secure environment 503 from the rest of the world. It prevents information and processes within the secure environment 503 from being observed, interfered with and leaving except under appropriate secure conditions. Barrier 502 also controls external access to secure resources, processes and information within SPU 500. In one example, tamper resistant security barrier 502 is formed by security features such as "encryption," and hardware that detects tampering and/or destroys sensitive information within secure environment 503 when tampering is detected.

SPU 500 in this example is an integrated circuit ("IC") "chip" 504 including "hardware" 506 and "firmware" 508. SPU 500 connects to the rest of the electronic appliance through an "appliance link" 510. SPU "firmware" 508 in this example is "software" such as a "computer program(s)" "embedded" within

chip 504. Firmware 508 makes the hardware 506 work.
Hardware 506 preferably contains a processor to perform
instructions specified by firmware 508. "Hardware" 506 also
contains long-term and short-term memories to store information
5 securely so it can't be tampered with. SPU 500 may also have a
protected clock/calendar used for timing events. The SPU
hardware 506 in this example may include special purpose
electronic circuits that are specially designed to perform certain
processes (such as "encryption" and "decryption") rapidly and
10 efficiently.

The particular context in which SPU 500 is being used will
determine how much processing capabilities SPU 500 should
have. SPU hardware 506, in this example, provides at least
15 enough processing capabilities to support the secure parts of
processes shown in Figure 3. In some contexts, the functions of
SPU 500 may be increased so the SPU can perform all the
electronic appliance processing, and can be incorporated into a
general purpose processor. In other contexts, SPU 500 may work
20 alongside a general purpose processor, and therefore only needs
to have enough processing capabilities to handle secure
processes.

VDE Electronic Appliance and Rights Operating System*

Figure 7 shows an example of an electronic appliance 600 including SPU 500. Electronic appliance 600 may be practically any kind of electrical or electronic device, such as:

5

- a computer
- a T.V. "set top" control box
- a pager
- a telephone
- a sound system
- a video reproduction system
- a video game player
- a "smart" credit card

10

15

Electronic appliance 600 in this example may include a keyboard or keypad 612, a voice recognizer 613, and a display 614. A human user can input commands through keyboard 612 and/or voice recognizer 613, and may view information on display 614.

20

Appliance 600 may communicate with the outside world through any of the connections/devices normally used within an electronic appliance. The connections/devices shown along the bottom of the drawing are examples:

a "modem" 618 or other telecommunications link;

a CD ROM disk 620 or other storage medium or device;

a printer 622;
broadcast reception 624;
a document scanner 626; and
a "cable" 628 connecting the appliance with a "network."

5

Virtual distribution environment 100 provides a "rights operating system" 602 that manages appliance 600 and SPU 500 by controlling their hardware resources. The operating system 602 may also support at least one "application" 608. Generally,
10 "application" 608 is hardware and/or software specific to the context of appliance 600. For example, if appliance 600 is a personal computer, then "application" 608 could be a program loaded by the user, for instance, a word processor, a communications system or a sound recorder. If appliance 600 is
15 a television controller box, then application 608 might be hardware or software that allows a user to order videos on demand and perform other functions such as fast forward and rewind. In this example, operating system 602 provides a standardized, well defined, generalized "interface" that could
20 support and work with many different "applications" 608.

Operating system 602 in this example provides "rights and auditing operating system functions" 604 and "other operating system functions" 606. The "rights and auditing operating

system functions" 604 securely handle tasks that relate to virtual distribution environment 100. SPU 500 provides or supports many of the security functions of the "rights and auditing operating system functions" 402. The "other operating system functions" 606 handle general appliance functions. Overall operating system 602 may be designed from the beginning to include the "rights and auditing operating system functions" 604 plus the "other operating system functions" 606, or the "rights and auditing operating system functions" may be an add-on to a preexisting operating system providing the "other operating system functions."

"Rights operating system" 602 in this example can work with many different types of appliances 600. For example, it can work with large mainframe computers, "minicomputers" and "microcomputers" such as personal computers and portable computing devices. It can also work in control boxes on the top of television sets, small portable "pagers," desktop radios, stereo sound systems, telephones, telephone switches, or any other electronic appliance. This ability to work on big appliances as well as little appliances is called "scalable." A "scalable" operating system 602 means that there can be a standardized interface across many different appliances performing a wide variety of tasks.

The "rights operating system functions" 604 are "services-based" in this example. For example, "rights operating system functions" 604 handle summary requests from application 608 rather than requiring the application to always make more
5 detailed "subrequests" or otherwise get involved with the underlying complexities involved in satisfying a summary request. For example, application 608 may simply ask to read specified information; "rights operating system functions" 604 can then decide whether the desired information is VDE-
10 protected content and, if it is, perform processes needed to make the information available. This feature is called "transparency." "Transparency" makes tasks easy for the application 608. "Rights operating system functions" 604 can support applications 608 that "know" nothing about virtual distribution environment
15 100. Applications 608 that are "aware" of virtual distribution environment 100 may be able to make more detailed use of virtual distribution environment 100.

In this example, "rights operating system functions" 604
20 are "event driven". Rather than repeatedly examining the state of electronic appliance 600 to determine whether a condition has arisen, the "rights operating system functions" 604 may respond directly to "events" or "happenings" within appliance 600.

In this example, some of the services performed by "rights operating system functions" 604 may be extended based on additional "components" delivered to operating system 602. "Rights operating system functions" 604 can collect together and use "components" sent by different participants at different times. The "components" help to make the operating system 602 "scalable." Some components can change how services work on little appliances versus how they work on big appliances (e.g., multi-user). Other components are designed to work with specific applications or classes of applications (e.g., some types of meters and some types of budgets).

Electronic Appliance 600

An electronic appliance 600 provided by the preferred embodiment may, for example, be any electronic apparatus that contains one or more microprocessors and/or microcontrollers and/or other devices which perform logical and/or mathematical calculations. This may include computers; computer terminals; device controllers for use with computers; peripheral devices for use with computers; digital display devices; televisions; video and audio/video projection systems; channel selectors and/or decoders for use with broadcast and/or cable transmissions; remote control devices; video and/or audio recorders; media players including compact disc players, videodisc players and

tape players; audio and/or video amplifiers; virtual reality machines; electronic game players; multimedia players; radios; telephones; videophones; facsimile machines; robots; numerically controlled machines including machine tools and the like; and
5 other devices containing one or more microcomputers and/or microcontrollers and/or other CPUs, including those not yet in existence.

Figure 8 shows an example of an electronic appliance 600.
10 This example of electronic appliance 600 includes a system bus 653. In this example, one or more conventional general purpose central processing units ("CPUs") 654 are connected to bus 653. Bus 653 connects CPU(s) 654 to RAM 656, ROM 658, and I/O controller 660. One or more SPUs 500 may also be connected to
15 system bus 653. System bus 653 may permit SPU(s) 500 to communicate with CPU(s) 654, and also may allow both the CPU(s) and the SPU(s) to communicate (e.g., over shared address and data lines) with RAM 656, ROM 658 and I/O controller 660. A power supply 659 may provide power to SPU
20 500, CPU 654 and the other system components shown.

In the example shown, I/O controller 660 is connected to secondary storage device 652, a keyboard/display 612,614, a communications controller 666, and a backup storage device 668.

Backup storage device 668 may, for example, store information on mass media such as a tape 670, a floppy disk, a removable memory card, etc. Communications controller 666 may allow electronic appliance 600 to communicate with other electronic appliances via network 672 or other telecommunications links. Different electronic appliances 600 may interoperate even if they use different CPUs and different instances of ROS 602, so long as they typically use compatible communication protocols and/or security methods. In this example, I/O controller 660 permits CPU 654 and SPU 500 to read from and write to secondary storage 662, keyboard/display 612, 614, communications controller 666, and backup storage device 668.

Secondary storage 662 may comprise the same one or more non-secure secondary storage devices (such as a magnetic disk and a CD-ROM drive as one example) that electronic appliance 600 uses for general secondary storage functions. In some implementations, part or all of secondary storage 652 may comprise a secondary storage device(s) that is physically enclosed within a secure enclosure. However, since it may not be practical or cost-effective to physically secure secondary storage 652 in many implementations, secondary storage 652 may be used to store information in a secure manner by encrypting information before storing it in secondary storage 652. If information is

encrypted before it is stored, physical access to secondary storage 652 or its contents does not readily reveal or compromise the information.

5 Secondary storage 652 in this example stores code and data used by CPU 654 and/or SPU 500 to control the overall operation of electronic appliance 600. For example, Figure 8 shows that "Rights Operating System" ("ROS") 602 (including a portion 604 of ROS that provides VDE functions and a portion 10 606 that provides other OS functions) shown in Figure 7 may be stored on secondary storage 652. Secondary storage 652 may also store one or more VDE objects 300. Figure 8 also shows that the secure files 610 shown in Figure 7 may be stored on secondary storage 652 in the form of a "secure database" or 15 management file system 610. This secure database 610 may store and organize information used by ROS 602 to perform VDE functions 604. Thus, the code that is executed to perform VDE and other OS functions 604, 606, and secure files 610 (as well as VDE objects 300) associated with those functions may be stored 20 in secondary storage 652. Secondary storage 652 may also store "other information" 673 such as, for example, information used by other operating system functions 606 for task management, non-VDE files, etc. Portions of the elements indicated in secondary storage 652 may also be stored in ROM 658, so long as

those elements do not require changes (except when ROM 658 is replaced). Portions of ROS 602 in particular may desirably be included in ROM 658 (e.g., "bootstrap" routines, POST routines, etc. for use in establishing an operating environment for
5 electronic appliance 600 when power is applied).

Figure 8 shows that secondary storage 652 may also be used to store code ("application programs") providing user application(s) 608 shown in Figure 7. Figure 8 shows that there
10 may be two general types of application programs 608: "VDE aware" applications 608a, and Non-VDE aware applications 608b. VDE aware applications 608a may have been at least in part designed specifically with VDE 100 in mind to access and take detailed advantage of VDE functions 604. Because of the
15 "transparency" features of ROS 602, non-VDE aware applications 608b (e.g., applications not specifically designed for VDE 100) can also access and take advantage of VDE functions 604.

20 **SECURE PROCESSING UNIT 500**

Each VDE node or other electronic appliance 600 in the preferred embodiment may include one or more SPUs 500. SPUs 500 may be used to perform all secure processing for VDE 100. For example, SPU 500 is used for decrypting (or otherwise

unsecuring) VDE protected objects 300. It is also used for
managing encrypted and/or otherwise secured communication
(such as by employing authentication and/or error-correction
validation of information). SPU 500 may also perform secure
5 data management processes including governing usage of,
auditing of, and where appropriate, payment for VDE objects 300
(through the use of prepayments, credits, real-time electronic
debits from bank accounts and/or VDE node currency token
deposit accounts). SPU 500 may perform other transactions
10 related to such VDE objects 300.

SPU Physical Packaging and Security Barrier 502

As shown Figure 6, in the preferred embodiment, an SPU
500 may be implemented as a single integrated circuit "chip" 505
15 to provide a secure processing environment in which confidential
and/or commercially valuable information can be safely
processed, encrypted and/or decrypted. IC chip 505 may, for
example, comprise a small semiconductor "die" about the size of a
thumbnail. This semiconductor die may include semiconductor
20 and metal conductive pathways. These pathways define the
circuitry, and thus the functionality, of SPU 500. Some of these
pathways are electrically connected to the external "pins" 504 of
the chip 505.

As shown in Figures 6 and 9, SPU 500 may be surrounded by a tamper-resistant hardware security barrier 502. Part of this security barrier 502 is formed by a plastic or other package in which an SPU "die" is encased. Because the processing occurring within, and information stored by, SPU 500 are not easily accessible to the outside world, they are relatively secure from unauthorized access and tampering. All signals cross barrier 502 through a secure, controlled path provided by BIU 530 that restricts the outside world's access to the internal components within SPU 500. This secure, controlled path resists attempts from the outside world to access secret information and resources within SPU 500.

It is possible to remove the plastic package of an IC chip and gain access to the "die." It is also possible to analyze and "reverse engineer" the "die" itself (e.g., using various types of logic analyzers and microprobes to collect and analyze signals on the die while the circuitry is operating, using acid etching or other techniques to remove semiconductor layers to expose other layers, viewing and photographing the die using an electron microscope, etc.) Although no system or circuit is absolutely impervious to such attacks, SPU barrier 502 may include additional hardware protections that make successful attacks exceedingly costly and time consuming. For example, ion

implantation and/or other fabrication techniques may be used to make it very difficult to visually discern SPU die conductive pathways, and SPU internal circuitry may be fabricated in such a way that it "self-destructs" when exposed to air and/or light.

5 SPU 500 may store secret information in internal memory that loses its contents when power is lost. Circuitry may be incorporated within SPU 500 that detects microprobing or other tampering, and self-destructs (or destroys other parts of the SPU) when tampering is detected. These and other hardware-

10 based physical security techniques contribute to tamper-resistant hardware security barrier 502.

To increase the security of security barrier 502 even further, it is possible to encase or include SPU 500 in one or

15 more further physical enclosures such as, for example: epoxy or other "potting compound"; further module enclosures including additional self-destruct, self-disabling or other features activated when tampering is detected; further modules providing additional security protections such as requiring password or

20 other authentication to operate; and the like. In addition, further layers of metal may be added to the die to complicate acid etching, micro probing, and the like; circuitry designed to "zeroize" memory may be included as an aspect of self-destruct processes; the plastic package itself may be designed to resist

chemical as well as physical "attacks"; and memories internal to SPU 500 may have specialized addressing and refresh circuitry that "shuffles" the location of bits to complicate efforts to electrically determine the value of memory locations. These and
5 other techniques may contribute to the security of barrier 502.

In some electronic appliances 600, SPU 500 may be integrated together with the device microcontroller or equivalent or with a device I/O or communications microcontroller into a
10 common chip (or chip set) 505. For example, in one preferred embodiment, SPU 500 may be integrated together with one or more other CPU(s) (e.g., a CPU 654 of an electronic appliance) in a single component or package. The other CPU(s) 654 may be any centrally controlling logic arrangement, such as for example,
15 a microprocessor, other microcontroller, and/or array or other parallel processor. This integrated configuration may result in lower overall cost, smaller overall size, and potentially faster interaction between an SPU 500 and a CPU 654. Integration may also provide wider distribution if an integrated SPU/CPU
20 component is a standard feature of a widely distributed microprocessor line. Merging an SPU 500 into a main CPU 654 of an electronic appliance 600 (or into another appliance or appliance peripheral microcomputer or other microcontroller) may substantially reduce the overhead cost of implementing

VDE 100. Integration considerations may include cost of implementation, cost of manufacture, desired degree of security, and value of compactness.

5 SPU 500 may also be integrated with devices other than CPUs. For example, for video and multimedia applications, some performance and/or security advantages (depending on overall design) could result from integrating an SPU 500 into a video controller chip or chipset. SPU 500 can also be integrated
10 directly into a network communications chip or chipset or the like. Certain performance advantages in high speed communications applications may also result from integrating an SPU 500 with a modem chip or chipset. This may facilitate incorporation of an SPU 500 into communication appliances such
15 as stand-alone fax machines. SPU 500 may also be integrated into other peripheral devices, such as CD-ROM devices, set-top cable devices, game devices, and a wide variety of other electronic appliances that use, allow access to, perform transactions related to, or consume, distributed information.

20

SPU 500 Internal Architecture

Figure 9 is a detailed diagram of the internal structure within an example of SPU 500. SPU 500 in this example includes a single microprocessor 520 and a limited amount of

memory configured as ROM 532 and RAM 534. In more detail,
this example of SPU 500 includes microprocessor 520, an
encrypt/decrypt engine 522, a DMA controller 526, a real-time
clock 528, a bus interface unit ("BIU") 530, a read only memory
5 (ROM) 532, a random access memory (RAM) 534, and a memory
management unit ("MMU") 540. DMA controller 526 and MMU
540 are optional, but the performance of SPU 500 may suffer if
they are not present. SPU 500 may also include an optional
pattern matching engine 524, an optional random number
10 generator 542, an optional arithmetic accelerator circuit 544, and
optional compression/decompression circuit 546. A shared
address/data bus arrangement 536 may transfer information
between these various components under control of
microprocessor 520 and/or DMA controller 526. Additional or
15 alternate dedicated paths 538 may connect microprocessor 520 to
the other components (e.g., encrypt/decrypt engine 522 via line
538a, real-time clock 528 via line 538b, bus interface unit 530 via
line 538c, DMA controller via line 538d, and memory
management unit (MMU) 540 via line 538e).

20
The following section discusses each of these SPU
components in more detail.

Microprocessor 520

Microprocessor 520 is the "brain" of SPU 500. In this example, it executes a sequence of steps specified by code stored (at least temporarily) within ROM 532 and/or RAM 534.

5 Microprocessor 520 in the preferred embodiment comprises a dedicated central processing arrangement (e.g., a RISC and/or CISC processor unit, a microcontroller, and/or other central processing means or, less desirably in most applications, process specific dedicated control logic) for executing instructions stored
10 in the ROM 532 and/or other memory. Microprocessor 520 may be separate elements of a circuitry layout, or may be separate packages within a secure SPU 500.

In the preferred embodiment, microprocessor 520 normally
15 handles the most security sensitive aspects of the operation of electronic appliance 600. For example, microprocessor 520 may manage VDE decrypting, encrypting, certain content and/or appliance usage control information, keeping track of usage of VDE secured content, and other VDE usage control related
20 functions.

Stored in each SPU 500 and/or electronic appliance secondary memory 652 may be, for example, an instance of ROS 602 software, application programs 608, objects 300 containing

VDE controlled property content and related information, and management database 610 that stores both information associated with objects and VDE control information. ROS 602 includes software intended for execution by SPU microprocessor 520 for, in part, controlling usage of VDE related objects 300 by electronic appliance 600. As will be explained, these SPU programs include "load modules" for performing basic control functions. These various programs and associated data are executed and manipulated primarily by microprocessor 520.

Real Time Clock (RTC) 528

In the preferred embodiment, SPU 500 includes a real time clock circuit ("RTC") 528 that serves as a reliable, tamper resistant time base for the SPU. RTC 528 keeps track of time of day and date (e.g., month, day and year) in the preferred embodiment, and thus may comprise a combination calendar and clock. A reliable time base is important for implementing time based usage metering methods, "time aged decryption keys," and other time based SPU functions.

The RTC 528 must receive power in order to operate. Optimally, the RTC 528 power source could comprise a small battery located within SPU 500 or other secure enclosure. However, the RTC 528 may employ a power source such as an

externally located battery that is external to the SPU 500. Such
an externally located battery may provide relatively
uninterrupted power to RTC 528, and may also maintain as
non-volatile at least a portion of the otherwise volatile RAM 534
5 within SPU 500.

In one implementation, electronic appliance power supply
659 is also used to power SPU 500. Using any external power
supply as the only power source for RTC 528 may significantly
10 reduce the usefulness of time based security techniques unless,
at minimum, SPU 500 recognizes any interruption (or any
material interruption) of the supply of external power, records
such interruption, and responds as may be appropriate such as
disabling the ability of the SPU 500 to perform certain or all
15 VDE processes. Recognizing a power interruption may, for
example, be accomplished by employing a circuit which is
activated by power failure. The power failure sensing circuit
may power another circuit that includes associated logic for
recording one or more power fail events. Capacitor discharge
20 circuitry may provide the necessary temporary power to operate
this logic. In addition or alternatively, SPU 500 may from time
to time compare an output of RTC 528 to a clock output of a host
electronic appliance 600, if available. In the event a discrepancy
is detected, SPU 500 may respond as appropriate, including

recording the discrepancy and/or disabling at least some portion of processes performed by SPU 500 under at least some circumstances.

5 If a power failure and/or RTC 528 discrepancy and/or other event indicates the possibility of tampering, SPU 500 may automatically destroy, or render inaccessible without privileged intervention, one or more portions of sensitive information it stores, such as execution related information and/or encryption
10 key related information. To provide further SPU operation, such destroyed information would have to be replaced by a VDE clearinghouse, administrator and/or distributor, as may be appropriate. This may be achieved by remotely downloading update and/or replacement data and/or code. In the event of a
15 disabling and/or destruction of processes and/or information as described above, the electronic appliance 600 may require a secure VDE communication with an administrator, clearinghouse, and/or distributor as appropriate in order to reinitialize the RTC 528. Some or all secure SPU 500 processes
20 may not operate until then.

It may be desirable to provide a mechanism for setting and/or synchronizing RTC 528. In the preferred embodiment, when communication occurs between VDE electronic appliance

600 and another VDE appliance, an output of RTC 528 may be compared to a controlled RTC 528 output time under control of the party authorized to be "senior" and controlling. In the event of a discrepancy, appropriate action may be taken, including
5 resetting the RTC 528 of the "junior" controlled participant in the communication.

SPU Encrypt/Decrypt Engine 522

In the preferred embodiment, SPU encrypt/decrypt engine
10 522 provides special purpose hardware (e.g., a hardware state machine) for rapidly and efficiently encrypting and/or decrypting data. In some implementations, the encrypt/decrypt functions may be performed instead by microprocessor 520 under software control, but providing special purpose encrypt/decrypt hardware
15 engine 522 will, in general, provide increased performance. Microprocessor 520 may, if desired, comprise a combination of processor circuitry and dedicated encryption/decryption logic that may be integrated together in the same circuitry layout so as to, for example, optimally share one or more circuit elements.

20

Generally, it is preferable that a computationally efficient but highly secure "bulk" encryption/decryption technique should be used to protect most of the data and objects handled by SPU 500. It is preferable that an extremely secure

encryption/decryption technique be used as an aspect of authenticating the identity of electronic appliances 600 that are establishing a communication channel and securing any transferred permission, method, and administrative information.

5 In the preferred embodiment, the encrypt/decrypt engine 522 includes both a symmetric key encryption/decryption circuit (e.g. DES, Skipjack/Clipper, IDEA, RC-2, RC-4, etc.) and an antisymmetric (asymmetric) or Public Key ("PK") encryption/decryption circuit. The public/private key
10 encryption/decryption circuit is used principally as an aspect of secure communications between an SPU 500 and VDE administrators, or other electronic appliances 600, that is between VDE secure subsystems. A symmetric encryption/decryption circuit may be used for "bulk" encrypting
15 and decrypting most data stored in secondary storage 662 of electronic appliance 600 in which SPU 500 resides. The symmetric key encryption/decryption circuit may also be used for encrypting and decrypting content stored within VDE objects
300.

20

DES or public/private key methods may be used for all encryption functions. In alternate embodiments, encryption and decryption methods other than the DES and public/private key methods could be used for the various encryption related

functions. For instance, other types of symmetric encryption/decryption techniques in which the same key is used for encryption and decryption could be used in place of DES encryption and decryption. The preferred embodiment can support a plurality of decryption/encryption techniques using multiple dedicated circuits within encrypt/decrypt engine 522 and/or the processing arrangement within SPU 500.

Pattern Matching Engine 524

Optional pattern matching engine 524 may provide special purpose hardware for performing pattern matching functions. One of the functions SPU 500 may perform is to validate/authenticate VDE objects 300 and other items. Validation/authentication often involves comparing long data strings to determine whether they compare in a predetermined way. In addition, certain forms of usage (such as logical and/or physical (contiguous) relatedness of accessed elements) may require searching potentially long strings of data for certain bit patterns or other significant pattern related metrics. Although pattern matching can be performed by SPU microprocessor 520 under software control, providing special purpose hardware pattern matching engine 524 may speed up the pattern matching process.

Compression/Decompression Engine 546

An optional compression/decompression engine 546 may be provided within an SPU 500 to, for example, compress and/or decompress content stored in, or released from, VDE objects 300.

5. Compression/decompression engine 546 may implement one or more compression algorithms using hardware circuitry to improve the performance of compression/decompression operations that would otherwise be performed by software operating on microprocessor 520, or outside SPU 500.

10 Decompression is important in the release of data such as video and audio that is usually compressed before distribution and whose decompression speed is important. In some cases, information that is useful for usage monitoring purposes (such as record separators or other delimiters) is "hidden" under a
15 compression layer that must be removed before this information can be detected and used inside SPU 500.

Random Number Generator 542

20 Optional random number generator 542 may provide specialized hardware circuitry for generating random values (e.g., from inherently unpredictable physical processes such as quantum noise). Such random values are particularly useful for constructing encryption keys or unique identifiers, and for initializing the generation of pseudo-random sequences.

Random number generator 542 may produce values of any convenient length, including as small as a single bit per use. A random number of arbitrary size may be constructed by concatenating values produced by random number generator 542. A cryptographically strong pseudo-random sequence may be generated from a random key and seed generated with random number generator 542 and repeated encryption either with the encrypt/decrypt engine 522 or cryptographic algorithms in SPU 500. Such sequences may be used, for example, in private headers to frustrate efforts to determine an encryption key through cryptanalysis.

Arithmetic Accelerator 544

An optional arithmetic accelerator 544 may be provided within an SPU 500 in the form of hardware circuitry that can rapidly perform mathematical calculations such as multiplication and exponentiation involving large numbers. These calculations can, for example, be requested by microprocessor 520 or encrypt/decrypt engine 522, to assist in the computations required for certain asymmetric encryption/decryption operations. Such arithmetic accelerators are well-known to those skilled in the art. In some implementations, a separate arithmetic accelerator 544 may be

omitted and any necessary calculations may be performed by microprocessor 520 under software control.

DMA Controller 526

5 DMA controller 526 controls information transfers over address/data bus 536 without requiring microprocessor 520 to process each individual data transfer. Typically, microprocessor 520 may write to DMA controller 526 target and destination addresses and the number of bytes to transfer, and DMA
10 controller 526 may then automatically transfer a block of data between components of SPU 500 (e.g., from ROM 532 to RAM 534, between encrypt/decrypt engine 522 and RAM 534, between bus interface unit 530 and RAM 534, etc.). DMA controller 526 may have multiple channels to handle multiple transfers
15 simultaneously. In some implementations, a separate DMA controller 526 may be omitted, and any necessary data movements may be performed by microprocessor 520 under software control.

20 Bus Interface Unit (BIU) 530

Bus interface unit (BIU) 530 communicates information between SPU 500 and the outside world across the security barrier 502. BIU 530 shown in Figure 9 plus appropriate driver software may comprise the "appliance link" 510 shown in Figure

6. Bus interface unit 530 may be modelled after a USART or PCI bus interface in the preferred embodiment. In this example, BIU 530 connects SPU 500 to electronic appliance system bus 653 shown in Figure 8. BIU 530 is designed to prevent unauthorized access to internal components within SPU 500 and their contents. It does this by only allowing signals associated with an SPU 500 to be processed by control programs running on microprocessor 520 and not supporting direct access to the internal elements of an SPU 500.

10

Memory Management Unit 540

Memory Management Unit (MMU) 540, if present, provides hardware support for memory management and virtual memory management functions. It may also provide heightened security by enforcing hardware compartmentalization of the secure execution space (e.g., to prevent a less trusted task from modifying a more trusted task). More details are provided below in connection with a discussion of the architecture of a Secure Processing Environment ("SPE") 503 supported by SPU 500.

15
20

MMU 540 may also provide hardware-level support functions related to memory management such as, for example, address mapping.

SPU Memory Architecture

In the preferred embodiment, SPU 500 uses three general kinds of memory:

(1) internal ROM 532;

5 (2) internal RAM 534; and

(3) external memory (typically RAM and/or disk supplied by a host electronic appliance).

10 The internal ROM 532 and RAM 534 within SPU 500 provide a secure operating environment and execution space. Because of cost limitations, chip fabrication size, complexity and other limitations, it may not be possible to provide sufficient memory within SPU 500 to store all information that an SPU needs to process in a secure manner. Due to the practical limits
15 on the amount of ROM 532 and RAM 534 that may be included within SPU 500, SPU 500 may store information in memory external to it, and move this information into and out of its secure internal memory space on an as needed basis. In these cases, secure processing steps performed by an SPU typically
20 must be segmented into small, securely packaged elements that may be "paged in" and "paged out" of the limited available internal memory space. Memory external to an SPU 500 may not be secure. Since the external memory may not be secure, SPU 500 may encrypt and cryptographically seal code and other

information before storing it in external memory. Similarly, SPU 500 must typically decrypt code and other information obtained from external memory in encrypted form before processing (e.g., executing) based on it. In the preferred embodiment, there are two general approaches used to address potential memory limitations in a SPU 500. In the first case, the small, securely packaged elements represent information contained in secure database 610. In the second case, such elements may represent protected (e.g., encrypted) virtual memory pages. Although virtual memory pages may correspond to information elements stored in secure database 610, this is not required in this example of a SPU memory architecture.

The following is a more detailed discussion of each of these three SPU memory resources.

SPU Internal ROM

SPU 500 read only memory (ROM) 532 or comparable purpose device provides secure internal non-volatile storage for certain programs and other information. For example, ROM 532 may store "kernel" programs such as SPU control firmware 508 and, if desired, encryption key information and certain fundamental "load modules." The "kernel" programs, load module information, and encryption key information enable the

control of certain basic functions of the SPU 500. Those components that are at least in part dependent on device configuration (e.g., POST, memory allocation, and a dispatcher) may be loaded in ROM 532 along with additional load modules that have been determined to be required for specific installations or applications.

In the preferred embodiment, ROM 532 may comprise a combination of a masked ROM 532a and an EEPROM and/or equivalent "flash" memory 532b. EEPROM or flash memory 532b is used to store items that need to be updated and/or initialized, such as for example, certain encryption keys. An additional benefit of providing EEPROM and/or flash memory 532b is the ability to optimize any load modules and library functions persistently stored within SPU 500 based on typical usage at a specific site. Although these items could also be stored in NVRAM 534b, EEPROM and/or flash memory 532b may be more cost effective.

Masked ROM 532a may cost less than flash and/or EEPROM 532b, and can be used to store permanent portions of SPU software/firmware. Such permanent portions may include, for example, code that interfaces to hardware elements such as the RTC 528, encryption/decryption engine 522, interrupt

handlers, key generators, etc. Some of the operating system,
library calls, libraries, and many of the core services provided by
SPU 500 may also be in masked ROM 532a. In addition, some of
the more commonly used executables are also good candidates for
5 inclusion in masked ROM 532a. Items that need to be updated
or that need to disappear when power is removed from SPU 500
should not be stored in masked ROM 532a.

Under some circumstances, RAM 534a and/or NVRAM
10 534b (NVRAM 534b may, for example, be constantly powered
conventional RAM) may perform at least part of the role of ROM
532.

SPU Internal RAM

15 SPU 500 general purpose RAM 534 provides, among other
things, secure execution space for secure processes. In the
preferred embodiment, RAM 534 is comprised of different types
of RAM such as a combination of high-speed RAM 534a and an
NVRAM ("non-volatile RAM") 534b. RAM 534a may be volatile,
20 while NVRAM 534b is preferably battery backed or otherwise
arranged so as to be non-volatile (i.e., it does not lose its contents
when power is turned off).

High-speed RAM 534a stores active code to be executed and associated data structures.

5 NVRAM 534b preferably contains certain keys and summary values that are preloaded as part of an initialization process in which SPU 500 communicates with a VDE administrator, and may also store changeable or changing information associated with the operation of SPU 500. For security reasons, certain highly sensitive information (e.g.,
10 certain load modules and certain encryption key related information such as internally generated private keys) needs to be loaded into or generated internally by SPU 500 from time to time but, once loaded or generated internally, should never leave the SPU. In this preferred embodiment, the SPU 500
15 non-volatile random access memory (NVRAM) 534b may be used for securely storing such highly sensitive information. NVRAM 534b is also used by SPU 500 to store data that may change frequently but which preferably should not be lost in a power down or power fail mode.

20

NVRAM 534b is preferably a flash memory array, but may in addition or alternatively be electrically erasable programmable read only memory (EEPROM), static RAM (SRAM), bubble memory, three dimensional holographic or other

electro-optical memory, or the like, or any other writable (e.g., randomly accessible) non-volatile memory of sufficient speed and cost-effectiveness.

5 **SPU External Memory**

The SPU 500 can store certain information on memory devices external to the SPU. If available, electronic appliance 600 memory can also be used to support any device external portions of SPU 500 software. Certain advantages may be gained by allowing the SPU 500 to use external memory. As one example, memory internal to SPU 500 may be reduced in size by using non-volatile read/write memory in the host electronic appliance 600 such as a non-volatile portion of RAM 656 and/or ROM 658.

15

Such external memory may be used to store SPU programs, data and/or other information. For example, a VDE control program may be, at least in part, loaded into the memory and communicated to and decrypted within SPU 500 prior to execution. Such control programs may be re-encrypted and communicated back to external memory where they may be stored for later execution by SPU 500. "Kernel" programs and/or some or all of the non-kernel "load modules" may be stored by SPU 500 in memory external to it. Since a secure database 610

20

may be relatively large, SPU 500 can store some or all of secure database 610 in external memory and call portions into the SPU 500 as needed.

5 As mentioned above, memory external to SPU 500 may not be secure. Therefore, when security is required, SPU 500 must encrypt secure information before writing it to external memory, and decrypt secure information read from external memory before using it. Inasmuch as the encryption layer relies on
10 secure processes and information (e.g., encryption algorithms and keys) present within SPU 500, the encryption layer effectively "extends" the SPU security barrier 502 to protect information the SPU 500 stores in memory external to it.

15 SPU 500 can use a wide variety of different types of external memory. For example, external memory may comprise electronic appliance secondary storage 652 such as a disk; external EEPROM or flash memory 658; and/or external RAM
20 656. External RAM 656 may comprise an external nonvolatile (e.g. constantly powered) RAM and/or cache RAM.

Using external RAM local to SPU 500 can significantly improve access times to information stored externally to an SPU. For example, external RAM may be used:

- to buffer memory image pages and data structures prior to their storage in flash memory or on an external hard disk (assuming transfer to flash or hard disk can occur in significant power or system failure cases);
- 5 • provide encryption and decryption buffers for data being released from VDE objects 300.
- to cache "swap blocks" and VDE data structures currently in use as an aspect of providing a secure virtual memory environment for SPU 500.
- 10 • to cache other information in order to, for example, reduce frequency of access by an SPU to secondary storage 652 and/or for other reasons.

Dual ported external RAM can be particularly effective in improving SPU 500 performance, since it can decrease the data
15 movement overhead of the SPU bus interface unit 530 and SPU microprocessor 520.

Using external flash memory local to SPU 500 can be used to significantly improve access times to virtually all data
20 structures. Since most available flash storage devices have limited write lifetimes, flash storage needs to take into account the number of writes that will occur during the lifetime of the flash memory. Hence, flash storage of frequently written temporary items is not recommended. If external RAM is non-

volatile, then transfer to flash (or hard disk) may not be necessary.

External memory used by SPU 500 may include two categories:

5

- external memory dedicated to SPU 500, and
- memory shared with electronic appliance 600.

10

For some VDE implementations, sharing memory (e.g., electronic appliance RAM 656, ROM 658 and/or secondary storage 652) with CPU 654 or other elements of an electronic appliance 600 may be the most cost effective way to store VDE secure database management files 610 and information that needs to be stored external to SPU 500. A host system hard disk secondary memory 652 used for general purpose file storage can, for example, also be used to store VDE management files 610. SPU 500 may be given exclusive access to the external memory (e.g., over a local bus high speed connection provided by BIU 530). Both dedicated and shared external memory may be provided.

15

20

SPU Integrated Within CPU

As discussed above, it may be desirable to integrate CPU 654 and SPU 500 into the same integrated circuit and/or device.

SPU 500 shown in Figure 9 includes a microprocessor 520 that may be similar or identical to a standard microprocessor available off-the-shelf from a variety of manufacturers. Similarly, the SPU DMA controller 526 and certain other

5 microprocessor support circuitry may be standard implementations available in off-the-shelf microprocessor and/or microcomputer chips. Since many of the general control and processing requirements provided by SPU 500 in the preferred embodiment can be satisfied using certain generic CPU and/or

10 microcontroller components, it may be desirable to integrate SPU VDE functionality into a standard generic CPU or microcontroller chip. Such an integrated solution can result in a very cost-effective "dual mode" component that is capable of performing all of the generic processing of a standard CPU as

15 well as the secure processing of an SPU. Many of the control logic functions performed by the preferred embodiment SPU can be performed by generic CPU and/or micro-controller logic so that at least a portion of the control logic does not have to be duplicated. Additional cost savings (e.g., in terms of reducing

20 manufacturing costs, inventory costs and printed circuit board real estate requirements) may also be obtained by not requiring an additional, separate physical SPU 500 device or package. Figure 9A shows one example architecture of a combination CPU/SPU 2650. CPU/SPU 2650 may include a standard

microprocessor or microcontroller 2652, a standard bus interface unit (BIU) 2656, and a standard (optional) DMA controller 2654, as well as various other standard I/O controllers, computation circuitry, etc. as may be found in a typical off-the-shelf

5 microprocessor/microcontroller. Real time clock 528 may be added to the standard architecture to give the CPU/SPU 2650 access to the real time clock functions as discussed above in connection with Figure 9. Real-time clock 528 must be protected from tampering in order to be secure. Such protections may

10 include internal or external backup power, an indication that its power (and thus its operation) has been interrupted, and/or an indication that the external clock signal(s) from which it derives its timing have been interfered with (e.g., sped up, slowed down). Similarly, an encrypt/decrypt engine 522, pattern matching

15 engine 524, compression/decompression engine 546 and/or arithmetic accelerator 544 may be added if desired to provide greater efficiencies, or the functions performed by these components could be provided instead by software executing on microprocessor 2652. An optional memory management unit 540

20 may also be provided if desired. A true random number generator 542 may be provided also if desired. Connections shown between mode interface switch 2658 and other components can carry both data and control information, specifically control information that determines what security-

relevant aspects of the other components are available for access and/or manipulation.

5 In addition, secure ROM 532 and/or secure RAM 534 may be provided within CPU/SPU 2650 along with a "mode interface switch" 2658a, 2658b. Mode interface switch 2658 selectively provides microprocessor 2652 with access to secure memory 532, 534 and other secure components (blocks 522, 546, 524, 542, 544, 528) depending upon the "mode" CPU/SPU 2650 is operating in.

10 CPU/SPU 2650 in this example may operate in two different modes:

- an "SPU" mode, or
- a "normal" mode.

In the "normal" mode, CPU/SPU 2650 operates

15 substantially identically to a standard off-the-shelf CPU while also protecting the security of the content, state, and operations of security-relevant components included in CPU/SPU 2650. Such security-relevant components may include the secure memories 532, 534; the encrypt/decrypt engine 522, the optional

20 pattern-matching engine 524, random number generator 542, arithmetic accelerator 544, the SPU-not-initialized flag 2671, the secure mode interface switch 2658, the real-time clock 528, the DMA controller 2654, the MMU 540, compress/decompress block

546, and/or any other components that may affect security of the operation of the CPU/SPU in "SPU" mode.

5 In this example, CPU/SPU 2650 operating in the "normal" mode controls mode interface switch 2658 to effectively "disconnect" (i.e., block unsecure access to) the security-relevant components, or to the security-relevant aspects of the operations of such components as have a function for both "normal" and "SPU" mode. In the "normal" mode, for example, microprocessor 10 2652 could access information from standard registers or other internal RAM and/or ROM (not shown), execute instructions in a "normal" way, and perform any other tasks as are provided within a standard CPU -- but could not access or compromise the contents of secure memory 532, 534 or access blocks 522, 524, 15 542, 544, 546. In this example "normal" mode, mode interface switch 2658 would effectively prevent any access (e.g., both read and write access) to secure memory 532, 534 so as to prevent the information stored within that secure memory from being compromised.

20

When CPU/SPU 2650 operates in the "SPU" mode, mode interface switch 2658 allows microprocessor 2652 to access secure memory 532, 534, and to control security-relevant aspects of other components in the CPU/SPU. The "SPU" mode in this

example requires all instructions executed by microprocessor 2652 to be fetched from secure memory 532, 534 -- preventing execution based on "mixed" secure and non-secure instructions. In the "SPU" mode, mode interface switch 2658 may, in one example embodiment, disconnect or otherwise block external accesses carried over bus 652 from outside CPU/SPU 2650 (e.g., DMA accesses, cache coherency control accesses) to ensure that the microprocessor 2652 is controlled entirely by instructions carried within or derived from the secure memory 532, 534.

Mode interface switch 2658 may also disconnect or otherwise block access by microprocessor 2652 to some external memory and/or other functions carried over bus 652. Mode interface switch 2658 in this example prevents other CPU operations/instructions from exposing the contents of secure memory 532, 534.

In the example shown in Figure 9A, the mode control of mode interface switch 2658 is based on a "mode" control signal provided by microprocessor 2652. In this example, microprocessor 2652 may be slightly modified so it can execute two "new" instructions:

- "enable 'SPU' mode" instruction, and
- "disable 'SPU' mode" instruction.

5 When microprocessor 2652 executes the "enable 'SPU'
mode" instruction, it sends an appropriate "mode" control signal
to mode interface switch 2658 to "switch" the interface switch
into the "SPU" mode of operation. When microprocessor 2652
executes the "disable 'SPU' mode" instruction, it sends an
appropriate "mode" control signal to mode interface switch 2658
10 to disable the "SPU" mode of operation.

When CPU/SPU 2650 begins operating in the "SPU" mode
(based on microprocessor 2652 executing the "enable "SPU"
mode" instruction), mode interface switch 2658 forces
15 microprocessor 2652 to begin fetching instructions from secure
memory 532, 534 (e.g., beginning at some fixed address) in one
example. When CPU/SPU 2650 begins operating in this example
"SPU" mode, mode interface switch 2658 may force
microprocessor 2652 to load its registers from some fixed address
20 in secure memory 532, 534 and may begin execution based on
such register content. Once operating in the "SPU" mode,
microprocessor 2652 may provide encryption/decryption and
other control capabilities based upon the code and other content
of secure memory 532, 534 needed to provide the VDE

functionality of SPU 500 described above. For example,
microprocessor 2652 operating under control of information
within secure memory 532, 534 may read encrypted information
from bus 652 via bus interface unit 2656, write decrypted
5 information to the bus interface unit, and meter and limit
decryption of such information based on values stored in the
secure memory.

At the end of secure processing, execution by
10 microprocessor 2652 of the "disable SPU mode" instruction may
cause the contents of all registers and other temporary storage
locations used by microprocessor 2652 that are not within secure
memory 532, 534 to be destroyed or copied into secure memory
532, 534 before "opening" mode interface switch 2658. Once
15 mode interface switch 2658 is "open," the microprocessor 2652 no
longer has access to secure memory 532, 534 or the information
it contained, or to control or modify the state of any other
security-relevant components or functions contained within
CPU/SPU 2650 to which access is controlled by mode interface
20 switch 2658.

Whenever CPU/SPU 2650 enters or leaves the "SPU"
mode, the transition is performed in such a way that no
information contained in the secure memory 532, 534 or derived

from it (e.g., stored in registers or a cache memory associated with microprocessor 2652) while in the "SPU" mode can be exposed by microprocessor 2652 operations that occur in the "normal" mode. This may be accomplished either by hardware mechanisms that protect against such exposure, software instructions executed in "SPU" mode that clear, reinitialize, and otherwise reset during such transitions, or a combination of both.

In some example implementations, interrupts may be enabled while CPU/SPU 2650 is operating in the "SPU" mode similarly interrupts and returns from interrupts while in the "SPU" mode may allow transitions from "SPU" mode to "normal" mode and back to "SPU" mode without exposing the content of secure memory 532, 534 or the content of registers or other memory associated with microprocessor 2652 that may contain information derived from secure mode operation.

In some example implementations, there may be CPU/SPU activities such as DMA transfers between external memory and/or devices and secure memory 532, 534 that are initiated by microprocessor 2652 but involve autonomous activity by DMA controller 2654 and, optionally, encrypt/decrypt engine 522 and/or compress/decompress engine 546. In such implementations, mode interface switch 2658 and its associated

control signals may be configured to permit such pending activities (e.g. DMA transfers) to continue to completion even after CPU/SPU 2650 leaves "SPU" mode, provided that upon completion, all required clearing, reinitialization, and/or reset
5 activities occur, and provided that no access or interference is permitted with the pending activities except when CPU/SPU 2650 is operating in "SPU" mode.

In an additional example embodiment,
10 encryption/decryption logic may be connected between microprocessor 2652 and secure memory 532, 354. This additional encryption/decryption logic may be connected "in parallel" to mode interface switch 2658. The additional encryption/decryption logic may allow certain accesses by
15 microprocessor 2652 to the secure memory 532, 534 when CPU/SPU 2650 is operating in the "normal" mode. In this alternate embodiment, reads from secure memory 532, 534 when CPU/SPU 2650 is operating in the "normal" mode automatically result in the read information being encrypted before it is
20 delivered to microprocessor 2652 (and similarly, and writes to the secure memory may result in the written information being decrypted before it is deposited into the secure memory). This alternative embodiment may permit access to secure memory 532, 534 (which may in this example store the information in

"clear" form) by microprocessor 2652 when CPU/SPU 2650 is operating in the "non-secure normal" mode, but only reveals the secure memory contents to microprocessor 2652 in unencrypted form when the CPU/SPU is operating in the "SPU" mode. Such access may also be protected by cryptographic authentication techniques (e.g., message authentication codes) to prevent modification or replay attacks that modify encrypted data stored in secure memory 532, 534. Such protection may be performed utilizing either or both of software and/or hardware cryptographic techniques.

All of the components shown in Figure 9A may be disposed within a single integrated circuit package. Alternatively, mode interface switch 2658 and secure memory 532, 534, and other security-relevant components might be placed within an integrated circuit chip package and/or other package separate from the rest of CPU/SPU 2650. In this two-package version, a private bus could be used to connect microprocessor 2652 to the mode interface switch 2658 and associated secure memory 532, 534. To maintain security in such multi-package versions, it may be necessary to enclose all the packages and their interconnections in an external physical tamper-resistant barrier.

Initialization of Integrated CPU/SPU

Instructions and/or data may need to be loaded into CPU/SPU 2650 before it can operate effectively as an SPU 500.

5 This may occur during the manufacture of CPU/SPU 2650 or subsequently at a CPU/SPU initialization facility. Security of such initialization may depend on physical control of access to the CPU/SPU component(s), on cryptographic means, or on some combination of both. Secure initialization may be performed in
10 plural steps under the control of different parties, such that an initialization step to be performed by party B is preconditioned on successful performance of a step by party A. Different initialization steps may be protected using different security techniques (e.g. physical access, cryptography).

15

In this example, switch 2658 may expose an external control signal 2670 that requests operation in "SPU" mode rather than "normal" mode after a power-on reset. This signal would be combined (e.g., by a logical AND 2672) with a non-volatile
20 storage element 2671 internal to CPU/SPU 2650. If both of these signals are asserted, AND gate 2672 would cause CPU/SPU 2650 to begin operating in SPU mode, either executing existing instructions from an address in SPU memory 532, executing instructions from main memory 2665 or otherwise external to the

CPU/SPU. The instructions thus executed would permit arbitrary initialization and other functions to be performed in "SPU" mode without necessarily requiring any instructions to be previously resident in the SPU memory 532.

5

Once initialized, the SPU would, under control of its initialization program, indicate to switch 2658 that the flag 2671 is to be cleared. Clearing flag 2671 would permanently disable this initialization capability because no mechanism would be provided to set flag 2671 back to its initial value.

10

If flag 2671 is clear, or control signal 2670 is not asserted, CPU/SPU 2650 would behave precisely as does microprocessor 2652 with respect to power-on reset and other external conditions. Under such conditions, only execution of the "enable SPU mode" instruction or otherwise requesting SPU mode under program control would cause "SPU" mode to be entered.

15

Additionally, a mechanism could be provided to permit microprocessor 2652 and/or control signal 2672 to reinitialize the flag 2671. Such reinitialization would be performed in a manner that cleared secure memory 532, 534 of any security-relevant information and reinitialized the state of all security-relevant components. This reinitialization mechanism would permit CPU/SPU 2650 to be initialized several times, facilitating testing

20

and/or re-use for different applications, while protecting all security-relevant aspects of its operation.

5 In the preferred embodiment, CPU/SPU 2650 would, when SPU mode has not yet been established, begin operating in SPU mode by fetching instructions from secure non-volatile memory 532, thereby ensuring a consistent initialization sequence and preventing SPU dependence on any information held outside CPU/SPU 2650. This approach permits secret initialization information (e.g., keys for validating digital signatures on 10 additional information to be loaded into secure memory 532, 534) to be held internally to CPU/SPU 2650 so that it is never exposed to outside access. Such information could even be supplied by a hardware "mask" used in the semiconductor fabrication process.

15

CPU/SPU Integrated With Unmodified Microprocessor

Figure 9B shows an additional example embodiment, in which a completely standard microprocessor 2652 integrated circuit chip could be transformed into a CPU/SPU 2650 by 20 adding an SPU chip 2660 that mediates access to external I/O devices and memory. In such an embodiment, the microprocessor 2652 would be connected to the SPU chip 2660 by a private memory bus 2661, and all three such components

would be contained within hardware tamper-resistant barrier
502.

5 In this embodiment, SPU chip 2660 may have the same
secure components as in Figure 9, i.e., it may have a
ROM/EEPROM 532, a RAM 532, an RTC 528, an (optional)
encryption/decryption engine 522, an (optional) random number
generator (RNG) 542, an (optional) arithmetic accelerator 544,
and a (optional) compression/decompression engine 546, and a
10 (optional) pattern matching circuit 524. Microprocessor 520 is
omitted from SPU chip 2660 since the standard microprocessor
2650 performs the processing functions instead. In addition,
SPU chip 2660 may include a flag 2671 and AND gate logic 2672
for the initialization purposes discussed above.

15

In addition, SPU chip 2660 includes an enhanced switch
2663 that provides the same overall (bus enhanced) functionality
performed by the switch 2658 in the Figure 9A embodiment.

20

Enhanced switch 2663 would perform the functions of a
bus repeater, mediator and interpreter. For example, enhanced
switch 2663 may act as a bus repeater that enables
microprocessor 2652's memory accesses made over internal
memory bus 2661 to be reflected to external memory bus 2664

and performed on main memory 2665. Enhanced switch 2663 may also act as a bus repeater similarly for internal I/O bus 2662 to external I/O bus 2665 in the event that microprocessor 2652 performs I/O operations distinctly from memory operations.

5 Enhanced switch 2663 may also perform the function of a mediator for microprocessor control functions 2666 (e.g., non-maskable interrupt, reset) with respect to externally requested control functions 2667. Enhanced switch 2663 may also provide mediation for access to SPU-protected resources

10 such as ROM 532, RAM 534, encrypt/decrypt engine 522 (if present), random number generator 542 (if present), arithmetic accelerator 544 (if present), pattern matching engine 524 (if present), and real-time clock 528 (if present). Enhanced switch 2663 may also act as an interpreter of control signals received

15 from microprocessor 2652 indicating entry to, exit from, and control of SPU mode.

Switch 2663 in this example recognizes a specific indication (e.g., an instruction fetch access to a designated

20 address in the secure memory 532) as the equivalent to the "enable 'SPU' mode" instruction. Upon recognizing such an indication, it may isolate the CPU/SPU 2650 from external buses and interfaces 2664, 2665, and 2667 such that any external activity, such as DMA cycles, would be "held" until the switch

2663 permits access again. After this, switch 2663 permits a single access to a specific location in secure memory 532 to complete.

5 The single instruction fetched from the designated location performs a control operation (a cache flush, for example), that can only be performed in microprocessor 2652's most privileged operating mode, and that has an effect visible to switch 2663. Switch 2663 awaits the occurrence of this event, and if it does
10 not occur within the expected number of cycles, does not enter "SPU" mode.

 Occurrence of the control operation demonstrates that microprocessor 2652 is executing in its most privileged "normal"
15 mode and therefore can be trusted to execute successfully the "enter 'SPU' mode" sequence of instructions stored in secure memory 532. If microprocessor 2652 were not executing in its most privileged mode, there would be no assurance that those instructions would execute successfully. Because switch 2663
20 isolates microprocessor 2652 from external signals (e.g., interrupts) until "SPU" mode is successfully initialized, the entry instructions can be guaranteed to complete successfully.

Following the initial instruction, switch 2663 can enter "partial SPU mode," in which a restricted area of ROM 532 and RAM 534 may be accessible. Subsequent instructions in secure memory 532 may then be executed by microprocessor 2652 to place it into a known state such that it can perform SPU functions -- saving any previous state in the restricted area of RAM 534 that is accessible. After the known state is established, an instruction may be executed to deliver a further indication (e.g., a reference to another designated memory location) to switch 2663, which would enter "SPU" mode. If this further indication is not received within the expected interval, switch 2663 will not enter "SPU" mode. Once in "SPU" mode, switch 2663 permits access to all of ROM 532, RAM 534, and other devices in SPU chip 2660.

15

The instructions executed during "partial SPU" mode must be carefully selected to ensure that no similar combination of instructions and processor state could result in a control transfer out of the protected SPU code in ROM 532 or RAM 534. For example, internal debugging features of microprocessor 2652 must be disabled to ensure that a malicious program could not set up a breakpoint later within protected SPU code and receive control. Similarly, all address translation must be disabled or reinitialized to ensure that previously created MMU data

20

structures would not permit SPU memory accesses to be
compromised. The requirement that the instructions for "partial
SPU mode" run in the microprocessor 2652's most privileged
mode is necessary to ensure that all its processor control
5 functions can be effectively disabled.

The switch 2663 provides additional protection against
tampering by ensuring that the expected control signals occur
after an appropriate number of clock cycles. Because the "partial
10 SPU" initialization sequence is entirely deterministic, it is not
feasible for malicious software to interfere with it and still retain
the same timing characteristics, even if malicious software is
running in microprocessor 2652's most privileged mode.

15 Once in "SPU" mode, switch 2663 may respond to
additional indications or signals generated by microprocessor
2652 (e.g., references to specific memory addresses) controlling
features of SPU mode. These might include enabling access to
external buses 2664 and 2665 so that SPU-protected code could
20 reference external memory or devices. Any attempts by
components outside CPU/SPU 2650 to perform operations (e.g.,
accesses to memory, interrupts, or other control functions) may
be prevented by switch 2663 unless they had been explicitly
enabled by instructions executed after "SPU" mode is entered.

To leave SPU mode and return to normal operation, the instructions executing in "SPU" mode may provide a specific indication to switch 2663 (e.g., a transfer to a designated memory address). This indication may be recognized by switch 2663 as
5 indicating a return to "normal mode," and it may again restrict access to ROM 532, RAM 534, and all other devices within SPU chip 2660, while re-enabling external buses and control lines 2664, 2665, and 2667. The instructions executed subsequently may restore the CPU state to that which was saved on entry to
10 SPU mode, so that microprocessor 2652 may continue to perform functions in progress when the SPU was invoked.

In an alternate embodiment, the entry into SPU mode may be conditioned on an indication recognized by switch 2663, but
15 the switch may then use a hardware mechanism (e.g., the processor's RESET signal) to reinitialize microprocessor 2562. In such an embodiment, switch 2663 may not implement partial SPU mode, but may instead enter SPU mode directly and ensure that the address from which instructions would be fetched by
20 microprocessor 2652 (specific to microprocessor 2652's architecture) results in accesses to appropriate locations in the SPU memory 532. This could reduce the complexity of the SPU mode entry mechanisms in switch 2663, but could incur an

additional processing cost from using a different reinitialization mechanism for microprocessor 2652.

5 SPU chip 2660 may be customized to operate in conjunction with a particular commercial microprocessor. In this example, the SPU may be customized to contain at least the specialized "enter SPU mode" instruction sequences to reinitialize the processor's state and, to recognize special indications for SPU control operations. SPU chip 2660 may also
10 be made electrically compatible with microprocessor 2652's external bus interfaces. This compatibility would permit CPU/SPU 2650 to be substituted for microprocessor 2652 without change either to software or hardware elsewhere in a computer system.

15 In other alternate embodiments, the functions described above for SPU chip 2600, microprocessor 2652, and internal buses 2661, 2662, and 2666 could all be combined within a single integrated circuit package, and/or on a single silicon die. This
20 could reduce packaging complexity and/or simplify establishment of the hardware tamper-resistant barrier 502.

* * * * *

The hardware configuration of an example of electronic appliance 600 has been described above. The following section describes an example of the software architecture of electronic appliance 600 provided by the preferred embodiment, including the structure and operation of preferred embodiment "Rights Operating System" ("ROS") 602.

Rights Operating System 602

Rights Operating System ("ROS") 602 in the preferred embodiment is a compact, secure, event-driven, services-based, "component" oriented, distributed multiprocessing operating system environment that integrates VDE information security control information, components and protocols with traditional operating system concepts. Like traditional operating systems, ROS 602 provided by the preferred embodiment is a piece of software that manages hardware resources of a computer system and extends management functions to input and/or output devices, including communications devices. Also like traditional operating systems, preferred embodiment ROS 602 provides a coherent set of basic functions and abstraction layers for hiding the differences between, and many of the detailed complexities of, particular hardware implementations. In addition to these characteristics found in many or most operating systems, ROS 602 provides secure VDE transaction management and other

advantageous features not found in other operating systems.

The following is a non-exhaustive list of some of the advantageous features provided by ROS 602 in the preferred embodiment:

5

Standardized interface provides coherent set of basic functions

- simplifies programming
- the same application can run on many different platforms

Event driven

10

- eases functional decomposition
- extendible
- accommodates state transition and/or process oriented

events

- simplifies task management

15

- simplifies inter-process communications

Services based

- allows simplified and transparent scalability
- simplifies multiprocessor support
- hides machine dependencies
- eases network management and support

20

Component Based Architecture

- processing based on independently deliverable secure components

- component model of processing control allows different sequential steps that are reconfigurable based on requirements
- components can be added, deleted or modified (subject to permissioning)
- full control information over pre-defined and user-defined application events
- events can be individually controlled with independent executables

10 Secure

- secure communications
- secure control functions
- secure virtual memory management
- information control structures protected from exposure
- data elements are validated, correlated and access controlled
- components are encrypted and validated independently
- components are tightly correlated to prevent unauthorized use of elements
- control structures and secured executables are validated prior to use to protect against tampering
- integrates security considerations at the I/O level
- provides on-the-fly decryption of information at release time

- enables a secure commercial transaction network
- flexible key management features

Scalaeble

- highly scalaeble across many different platforms
- 5 • supports concurrent processing in a multiprocessor environment
- supports multiple cooperating processors
- any number of host or security processors can be supported
- control structures and kernel are easily portable to various
- 10 host platforms and to different processors within a target platform without recompilation
- supports remote processing
- Remote Procedure Calls may be used for internal OS communications

15 Highly Integratable

- can be highly integrated with host platforms as an additional operating system layer
- permits non-secure storage of secured components and information using an OS layer "on top of" traditional OS
- 20 platforms
- can be seamlessly integrated with a host operating system to provide a common usage paradigm for transaction management and content access

- integration may take many forms: operating system layers for desktops (e.g., DOS, Windows, Macintosh); device drivers and operating system interfaces for network services (e.g, Unix and Netware); and dedicated component drivers for "low end" set tops are a few of many examples
- can be integrated in traditional and real time operating systems

Distributed

- provides distribution of control information and reciprocal control information and mechanisms
- supports conditional execution of controlled processes within any VDE node in a distributed, asynchronous arrangement
- controlled delegation of rights in a distributed environment
- supports chains of handling and control
- management environment for distributed, occasionally connected but otherwise asynchronous networked database
- real time and time independent data management
- supports "agent" processes

Transparent

- can be seamlessly integrated into existing operating systems

- can support applications not specifically written to use it

Network friendly

- internal OS structures may use RPCs to distribute processing
- 5 • subnets may seamlessly operate as a single node or independently

General Background Regarding Operating Systems

10 An "operating system" provides a control mechanism for organizing computer system resources that allows programmers to create applications for computer systems more easily. An operating system does this by providing commonly used functions, and by helping to ensure compatibility between different computer hardware and architectures (which may, for
15 example, be manufactured by different vendors). Operating systems also enable computer "peripheral device" manufacturers to far more easily supply compatible equipment to computer manufacturers and users.

20 Computer systems are usually made up of several different hardware components. These hardware components include, for example:

a central processing unit (CPU) for executing instructions;

an array of main memory cells (e.g., "RAM" or "ROM") for storing instructions for execution and data acted upon or parameterizing those instructions; and

5 one or more secondary storage devices (e.g., hard disk drive, floppy disk drive, CD-ROM drive, tape reader, card reader, or "flash" memory) organized to reflect named elements (a "file system") for storing images of main memory cells.

10 Most computer systems also include input/output devices such as keyboards, mice, video systems, printers, scanners and communications devices.

To organize the CPU's execution capabilities with
15 available RAM, ROM and secondary storage devices, and to provide commonly used functions for use by programmers, a piece of software called an "operating system" is usually included with the other components. Typically, this piece of software is designed to begin executing after power is applied to the
20 computer system and hardware diagnostics are completed. Thereafter, all use of the CPU, main memory and secondary memory devices is normally managed by this "operating system" software. Most computer operating systems also typically include a mechanism for extending their management functions

to I/O and other peripheral devices, including commonly used functions associated with these devices.

5 By managing the CPU, memory and peripheral devices through the operating system, a coherent set of basic functions and abstraction layers for hiding hardware details allows programmers to more easily create sophisticated applications. In addition, managing the computer's hardware resources with an operating system allows many differences in design and
10 equipment requirements between different manufacturers to be hidden. Furthermore, applications can be more easily shared with other computer users who have the same operating system, with significantly less work to support different manufacturers' base hardware and peripheral devices.

15

ROS 602 is an Operating System Providing Significant Advantages

ROS 602 is an "operating system." It manages the resources of electronic appliance 600, and provides a commonly
20 used set of functions for programmers writing applications 608 for the electronic appliance. ROS 602 in the preferred embodiment manages the hardware (e.g., CPU(s), memory(ies), secure RTC(s), and encrypt/decrypt engines) within SPU 500. ROS may also manage the hardware (e.g., CPU(s) and

memory(ies)) within one or more general purpose processors within electronic appliance 600. ROS 602 also manages other electronic appliance hardware resources, such as peripheral devices attached to an electronic appliance. For example, referring to Figure 7, ROS 602 may manage keyboard 612, display 614, modem 618, disk drive 620, printer 622, scanner 624. ROS 602 may also manage secure database 610 and a storage device (e.g., "secondary storage" 652) used to store secure database 610.

10

ROS 602 supports multiple processors. ROS 602 in the preferred embodiment supports any number of local and/or remote processors. Supported processors may include at least two types: one or more electronic appliance processors 654, and/or one or more SPUs 500. A host processor CPU 654 may provide storage, database, and communications services. SPU 500 may provide cryptographic and secured process execution services. Diverse control and execution structures supported by ROS 602 may require that processing of control information occur within a controllable execution space -- this controllable execution space may be provided by SPU 500. Additional host and/or SPU processors may increase efficiencies and/or capabilities. ROS 602 may access, coordinate and/or manage further processors remote to an electronic appliance 600 (e.g., via

15

20

network or other communications link) to provide additional processor resources and/or capabilities.

5 ROS 602 is services based. The ROS services provided
using a host processor 654 and/or a secure processor (SPU 500)
are linked in the preferred embodiment using a "Remote
Procedure Call" ("RPC") internal processing request structure.
Cooperating processors may request interprocess services using a
RPC mechanism, which is minimally time dependent and can be
10 distributed over cooperating processors on a network of hosts.
The multi-processor architecture provided by ROS 602 is easily
extensible to support any number of host or security processors.
This extensibility supports high levels of scalability. Services
also allow functions to be implemented differently on different
15 equipment. For example, a small appliance that typically has
low levels of usage by one user may implement a database
service using very different techniques than a very large
appliance with high levels of usage by many users. This is
another aspect of scalability.

20

ROS 602 provides a distributed processing environment.
For example, it permits information and control structures to
automatically, securely pass between sites as required to fulfill a
user's requests. Communications between VDE nodes under the

distributed processing features of ROS 602 may include
interprocess service requests as discussed above. ROS 602
supports conditional and/or state dependent execution of
controlled processors within any VDE node. The location that
5 the process executes and the control structures used may be
locally resident, remotely accessible, or carried along by the
process to support execution on a remote system.

ROS 602 provides distribution of control information,
10 including for example the distribution of control structures
required to permit "agents" to operate in remote environments.
Thus, ROS 602 provides facilities for passing execution and/or
information control as part of emerging requirements for "agent"
processes.

15 If desired, ROS 602 may independently distribute control
information over very low bandwidth connections that may or
may not be "real time" connections. ROS 602 provided by the
preferred embodiment is "network friendly," and can be
20 implemented with any level of networking protocol. Some
examples include e-mail and direct connection at approximately
"Layer 5" of the ISO model.

The ROS 602 distribution process (and the associated auditing of distributed information) is a controlled event that itself uses such control structures. This "reflective" distributed processing mechanism permits ROS 602 to securely distribute rights and permissions in a controlled manner, and effectively restrict the characteristics of use of information content. The controlled delegation of rights in a distributed environment and the secure processing techniques used by ROS 602 to support this approach provide significant advantages.

Certain control mechanisms within ROS 602 are "reciprocal." Reciprocal control mechanisms place one or more control components at one or more locations that interact with one or more components at the same or other locations in a controlled way. For example, a usage control associated with object content at a user's location may have a reciprocal control at a distributor's location that governs distribution of the usage control, auditing of the usage control, and logic to process user requests associated with the usage control. A usage control at a user's location (in addition to controlling one or more aspects of usage) may prepare audits for a distributor and format requests associated with the usage control for processing by a distributor. Processes at either end of a reciprocal control may be further controlled by other processes (e.g., a distributor may be limited

by a budget for the number of usage control mechanisms they may produce). Reciprocal control mechanisms may extend over many sites and many levels (e.g., a creator to a distributor to a user) and may take any relationship into account (e.g., creator/distributor, distributor/user, user/user, user/creator, user/creator/distributor, etc.) Reciprocal control mechanisms have many uses in VDE 100 in representing relationships and agreements in a distributed environment.

10 ROS 602 is scalable. Many portions of ROS 602 control structures and kernel(s) are easily portable to various host platforms without recompilation. Any control structure may be distributed (or redistributed) if a granting authority permits this type of activity. The executable references within ROS 602 are
15 portable within a target platform. Different instances of ROS 602 may execute the references using different resources. For example, one instance of ROS 602 may perform a task using an SPU 500, while another instance of ROS 602 might perform the same task using a host processing environment running in
20 protected memory that is emulating an SPU in software. ROS 602 control information is similarly portable; in many cases the event processing structures may be passed between machines and host platforms as easily as between cooperative processors in a single computer. Appliances with different levels of usage

and/or resources available for ROS 602 functions may implement those functions in very different ways. Some services may be omitted entirely if insufficient resources exist. As described elsewhere, ROS 602 "knows" what services are available, and
5 how to proceed based on any given event. Not all events may be processable if resources are missing or inadequate.

ROS 602 is component based. Much of the functionality provided by ROS 602 in the preferred embodiment may be based
10 on "components" that can be securely, independently deliverable, replaceable and capable of being modified (e.g., under appropriately secure conditions and authorizations). Moreover, the "components" may themselves be made of independently deliverable elements. ROS 602 may assemble these elements
15 together (using a construct provided by the preferred embodiment called a "channel") at execution time. For example, a "load module" for execution by SPU 500 may reference one or more "method cores," method parameters and other associated data structures that ROS 602 may collect and assemble together
20 to perform a task such as billing or metering. Different users may have different combinations of elements, and some of the elements may be customizable by users with appropriate authorization. This increases flexibility, allows elements to be reused, and has other advantages.

ROS 602 is highly secure. ROS 602 provides mechanisms to protect information control structures from exposure by end users and conduit hosts. ROS 602 can protect information, VDE control structures and control executables using strong encryption and validation mechanisms. These encryption and validation mechanisms are designed to make them highly resistant to undetected tampering. ROS 602 encrypts information stored on secondary storage device(s) 652 to inhibit tampering. ROS 602 also separately encrypts and validates its various components. ROS 602 correlates control and data structure components to prevent unauthorized use of elements. These features permit ROS 602 to independently distribute elements, and also allows integration of VDE functions 604 with non-secure "other" OS functions 606.

15

ROS 602 provided by the preferred embodiment extends conventional capabilities such as, for example, Access Control List (ACL) structures, to user and process defined events, including state transitions. ROS 602 may provide full control information over pre-defined and user-defined application events. These control mechanisms include "go/no-go" permissions, and also include optional event-specific executables that permit complete flexibility in the processing and/or controlling of events. This structure permits events to be

20

individually controlled so that, for example, metering and budgeting may be provided using independent executables. For example, ROS 602 extends ACL structures to control arbitrary granularity of information. Traditional operating systems provide static "go-no go" control mechanisms at a file or resource level; ROS 602 extends the control concept in a general way from the largest to the smallest sub-element using a flexible control structure. ROS 602 can, for example, control the printing of a single paragraph out of a document file.

ROS 602 provided by the preferred embodiment permits secure modification and update of control information governing each component. The control information may be provided in a template format such as method options to an end-user. An end-user may then customize the actual control information used within guidelines provided by a distributor or content creator. Modification and update of existing control structures is preferably also a controllable event subject to auditing and control information.

ROS 602 provided by the preferred embodiment validates control structures and secured executables prior to use. This validation provides assurance that control structures and executables have not been tampered with by end-users. The

validation also permits ROS 602 to securely implement components that include fragments of files and other operating system structures. ROS 602 provided by the preferred embodiment integrates security considerations at the operating system I/O level (which is below the access level), and provides "on-the-fly" decryption of information at release time. These features permit non-secure storage of ROS 602 secured components and information using an OS layer "on top of" traditional operating system platforms.

10

ROS 602 is highly integratable with host platforms as an additional operating system layer. Thus, ROS 602 may be created by "adding on" to existing operating systems. This involves hooking VDE "add ons" to the host operating system at the device driver and network interface levels. Alternatively, ROS 602 may comprise a wholly new operating system that integrates both VDE functions and other operating system functions.

15

Indeed, there are at least three general approaches to integrating VDE functions into a new operating system, potentially based on an existing operating system, to create a Rights Operating System 602 including:

20

- (1) Redesign the operating system based on VDE transaction management requirements;
- (2) Compile VDE API functions into an existing operating systems; and
- 5 (3) Integrate a VDE Interpreter into an existing operating system.

The first approach could be most effectively applied when a new operating system is being designed, or if a significant
10 upgrade to an existing operating system is planned. The transaction management and security requirements provided by the VDE functions could be added to the design requirements list for the design of a new operating system that provides, in an optimally efficient manner, an integration of "traditional"
15 operating system capabilities and VDE capabilities. For example, the engineers responsible for the design of the new version or instance of an operating system would include the requirements of VDE metering/transaction management in addition to other requirements (if any) that they use to form their design
20 approach, specifications, and actual implementations. This approach could lead to a "seamless" integration of VDE functions and capabilities by threading metering/transaction management functionality throughout the system design and implementation.

The second approach would involve taking an existing set of API (Application Programmer Interface) functions, and incorporating references in the operating system code to VDE function calls. This is similar to the way that the current

5 Windows operating system is integrated with DOS, wherein DOS serves as both the launch point and as a significant portion of the kernel underpinning of the Windows operating system. This approach would be also provide a high degree of "seamless" integration (although not quite as "seamless" as the first

10 approach). The benefits of this approach include the possibility that the incorporation of metering/transaction management functionality into the new version or instance of an operating system may be accomplished with lower cost (by making use of the existing code embodied in an API, and also using the design

15 implications of the API functional approach to influence the design of the elements into which the metering/transaction management functionality is incorporated).

The third approach is distinct from the first two in that it

20 does not incorporate VDE functionality associated with metering/transaction management and data security directly into the operating system code, but instead adds a new generalized capability to the operating system for executing metering/transaction management functionality. In this case, an

interpreter including metering/transaction management functions would be integrated with other operating system code in a "stand alone" mode. This interpreter might take scripts or other inputs to determine what metering/transaction management functions should be performed, and in what order and under which circumstances or conditions they should be performed.

Instead of (or in addition to) integrating VDE functions into/with an electronic appliance operating system, it would be possible to provide certain VDE functionality available as an application running on a conventional operating system.

ROS Software Architecture

Figure 10 is a block diagram of one example of a software structure/architecture for Rights Operating System ("ROS") 602 provided by the preferred embodiment. In this example, ROS 602 includes an operating system ("OS") "core" 679, a user Application Program Interface ("API") 682, a "redirector" 684, an "intercept" 692, a User Notification/Exception Interface 686, and a file system 687. ROS 602 in this example also includes one or more Host Event Processing Environments ("HPEs") 655 and/or one or more Secure Event Processing Environments ("SPEs") 503

(these environments may be generically referred to as "Protected Processing Environments" 650).

5 HPE(s) 655 and SPE(s) 503 are self-contained computing and processing environments that may include their own operating system kernel 688 including code and data processing resources. A given electronic appliance 600 may include any number of SPE(s) 503 and/or any number of HPE(s) 655. HPE(s) 655 and SPE(s) 503 may process information in a secure way,
10 and provide secure processing support for ROS 602. For example, they may each perform secure processing based on one or more VDE component assemblies 690, and they may each offer secure processing services to OS kernel 680.

15 In the preferred embodiment, SPE 503 is a secure processing environment provided at least in part by an SPU 500. Thus, SPU 500 provides the hardware tamper-resistant barrier 503 surrounding SPE 503. SPE 503 provided by the preferred embodiment is preferably:

- 20
- small and compact
 - loadable into resource constrained environments such as for example minimally configured SPUs 500
 - dynamically updatable

- extensible by authorized users
- integratable into object or procedural environments
- secure.

5

In the preferred embodiment, HPE 655 is a secure processing environment supported by a processor other than an SPU, such as for example an electronic appliance CPU 654 general-purpose microprocessor or other processing system or device. In the preferred embodiment, HPE 655 may be considered to "emulate" an SPU 500 in the sense that it may use software to provide some or all of the processing resources provided in hardware and/or firmware by an SPU. HPE 655 in one preferred embodiment of the present invention is full-featured and fully compatible with SPE 503—that is, HPE 655 can handle each and every service call SPE 503 can handle such that the SPE and the HPE are "plug compatible" from an outside interface standpoint (with the exception that the HPE may not provide as much security as the SPE).

20

HPEs 655 may be provided in two types: secure and not secure. For example, it may be desirable to provide non-secure versions of HPE 655 to allow electronic appliance 600 to efficiently run non-sensitive VDE tasks using the full resources

of a fast general purpose processor or computer. Such non-secure versions of HPE 655 may run under supervision of an instance of ROS 602 that also includes an SPE 503. In this way, ROS 602 may run all secure processes within SPE 503, and only
5 use HPE 655 for processes that do not require security but that may require (or run more efficiently) under potentially greater resources provided by a general purpose computer or processor supporting HPE 655. Non-secure and secure HPE 655 may operate together with a secure SPE 503.

10

HPEs 655 may (as shown in Figure 10) be provided with a software-based tamper resistant barrier 674 that makes them more secure. Such a software-based tamper resistant barrier 674 may be created by software executing on general-purpose
15 CPU 654. Such a "secure" HPE 655 can be used by ROS 602 to execute processes that, while still needing security, may not require the degree of security provided by SPU 500. This can be especially beneficial in architectures providing both an SPE 503 and an HPE 655. The SPU 502 may be used to perform all truly
20 secure processing, whereas one or more HPEs 655 may be used to provide additional secure (albeit possibly less secure than the SPE) processing using host processor or other general purpose resources that may be available within an electronic appliance 600. Any service may be provided by such a secure HPE 655. In

the preferred embodiment, certain aspects of "channel processing" appears to be a candidate that could be readily exported from SPE 503 to HPE 655.

5 The software-based tamper resistant barrier 674 provided by HPE 655 may be provided, for example, by: introducing time checks and/or code modifications to complicate the process of stepping through code comprising a portion of kernel 688a and/or a portion of component assemblies 690 using a debugger; using a
10 map of defects on a storage device (e.g., a hard disk, memory card, etc.) to form internal test values to impede moving and/or copying HPE 655 to other electronic appliances 600; using kernel code that contains false branches and other complications in flow of control to disguise internal processes to some degree from
15 disassembly or other efforts to discover details of processes; using "self-generating" code (based on the output of a co-sine transform, for example) such that detailed and/or complete instruction sequences are not stored explicitly on storage devices and/or in active memory but rather are generated as needed;
20 using code that "shuffles" memory locations used for data values based on operational parameters to complicate efforts to manipulate such values; using any software and/or hardware memory management resources of electronic appliance 600 to "protect" the operation of HPE 655 from other processes,

functions, etc. Although such a software-based tamper resistant barrier 674 may provide a fair degree of security, it typically will not be as secure as the hardware-based tamper resistant barrier 502 provided (at least in part) by SPU 500. Because security
5 may be better/more effectively enforced with the assistance of hardware security features such as those provided by SPU 500 (and because of other factors such as increased performance provided by special purpose circuitry within SPU 500), at least one SPE 503 is preferred for many or most higher security
10 applications. However, in applications where lesser security can be tolerated and/or the cost of an SPU 500 cannot be tolerated, the SPE 503 may be omitted and all secure processing may instead be performed by one or more secure HPEs 655 executing on general-purpose CPUs 654. Some VDE processes may not be
15 allowed to proceed on reduced-security electronic appliances of this type if insufficient security is provided for the particular process involved.

Only those processes that execute completely within SPEs
20 503 (and in some cases, HPEs 655) may be considered to be truly secure. Memory and other resources external to SPE 503 and HPEs 655 used to store and/or process code and/or data to be used in secure processes should only receive and handle that information in encrypted form unless SPE 503/HPE 655 can

protect secure process code and/or data from non-secure processes.

OS "core" 679 in the preferred embodiment includes a
5 kernel 680, an RPC manager 732, and an "object switch" 734.
API 682, HPE 655 and SPE 503 may communicate "event"
messages with one another via OS "core" 679. They may also
communicate messages directly with one another without
messages going through OS "core" 679.

10

Kernel 680 may manage the hardware of an electronic
appliance 600. For example, it may provide appropriate drivers
and hardware managers for interacting with input/output and/or
peripheral devices such as keyboard 612, display 614, other
15 devices such as a "mouse" pointing device and speech recognizer
613, modem 618, printer 622, and an adapter for network 672.
Kernel 680 may also be responsible for initially loading the
remainder of ROS 602, and may manage the various ROS tasks
(and associated underlying hardware resources) during
20 execution. OS kernel 680 may also manage and access secure
database 610 and file system 687. OS kernel 680 also provides
execution services for applications 608a(1), 608a(2), etc. and
other applications.

RPC manager 732 performs messaging routing and resource management/integration for ROS 680. It receives and routes "calls" from/to API 682, HPE 655 and SPE 503, for example.

5

Object switch 734 may manage construction, deconstruction and other manipulation of VDE objects 300.

10 User Notification/Exception Interface 686 in the preferred embodiment (which may be considered part of API 682 or another application coupled to the API) provides "pop up" windows/displays on display 614. This allows ROS 602 to communicate directly with a user without having to pass information to be communicated through applications 608. For
15 applications that are not "VDE aware," user notification/exception interface 686 may provide communications between ROS 602 and the user.

20 API 682 in the preferred embodiment provides a standardized, documented software interface to applications 608. In part, API 682 may translate operating system "calls" generated by applications 608 into Remote Procedure Calls ("RPCs") specifying "events." RPC manager 732 may route these RPCs to kernel 680 or elsewhere (e.g., to HPE(s) 655 and/or

SPE(s) 503, or to remote electronic appliances 600, processors, or VDE participants) for processing. The API 682 may also service RPC requests by passing them to applications 608 that register to receive and process specific requests.

5

API 682 provides an "Applications Programming Interface" that is preferably standardized and documented. It provides a concise set of function calls an application program can use to access services provided by ROS 602. In at least one preferred example, API 682 will include two parts: an application program interface to VDE functions 604; and an application program interface to other OS functions 606. These parts may be interwoven into the same software, or they may be provided as two or more discrete pieces of software (for example).

10

15

Some applications, such as application 608a(1) shown in Figure 11, may be "VDE aware" and may therefore directly access both of these parts of API 682. Figure 11A shows an example of this. A "VDE aware" application may, for example, include explicit calls to ROS 602 requesting the creation of new VDE objects 300, metering usage of VDE objects, storing information in VDE-protected form, etc. Thus, a "VDE aware" application can initiate (and, in some examples, enhance and/or extend) VDE functionality provided by ROS 602. In addition,

20

"VDE aware" applications may provide a more direct interface between a user and ROS 602 (e.g., by suppressing or otherwise dispensing with "pop up" displays otherwise provided by user notification/exception interface 686 and instead providing a more

5 "seamless" interface that integrates application and ROS messages).

Other applications, such as application 608b shown in Figure 11B, may not be "VDE Aware" and therefore may not

10 "know" how to directly access an interface to VDE functions 604 provided by API 682. To provide for this, ROS 602 may include a "redirector" 684 that allows such "non-VDE aware" applications 608(b) to access VDE objects 300 and functions 604. Redirector 684, in the preferred embodiment, translates OS calls directed to

15 the "other OS functions" 606 into calls to the "VDE functions" 604. As one simple example, redirector 684 may intercept a "file open" call from application 608(b), determine whether the file to be opened is contained within a VDE container 300, and if it is, generate appropriate VDE function call(s) to file system 687 to

20 open the VDE container (and potentially generate events to HPE 655 and/or SPE 503 to determine the name(s) of file(s) that may be stored in a VDE object 300, establish a control structure associated with a VDE object 300, perform a registration for a VDE object 300, etc.). Without redirector 684 in this example, a