Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2364625 C 2012/08/07

(11)(21) 2 364 625

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) Date de dépôt PCT/PCT Filing Date: 2000/03/23
(87) Date publication PCT/PCT Publication Date: 2000/09/28
(45) Date de délivrance/Issue Date: 2012/08/07
(85) Entrée phase nationale/National Entry: 2001/08/21
(86) N° demande PCT/PCT Application No.: US 2000/007780
(87) N° publication PCT/PCT Publication No.: 2000/057267
(30) Priorité/Priority: 1999/03/24 (US09/275,629)

(51) Cl.Int./Int.Cl. *G06F 3/06* (2006.01)

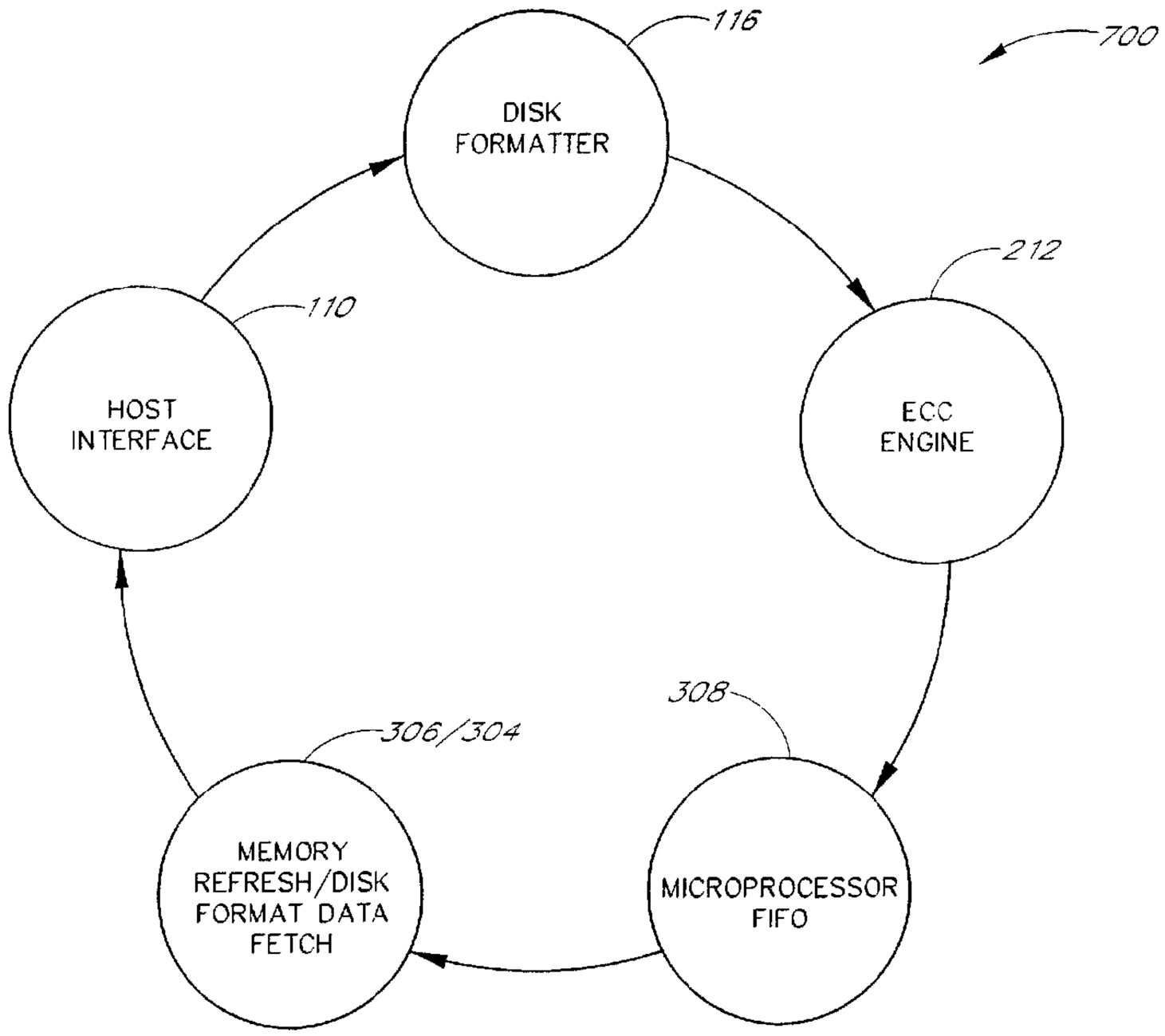
(72) Inventeurs/Inventors:
KRANTZ, LEON A., US;
CAMPBELL, FRANK W., US

(73) Propriétaire/Owner:
MARVELL WORLD TRADE LTD., BB

(74) Agent: SIM & MCBURNEY

(54) Titre : PROCEDE D'ARBITRAGE ET SYSTEMES D'ARBITRAGE DE L'ACCES A UNE MEMOIRE DE CONTROLEUR DE DISQUE DUR

(54) Title: ARBITRATION METHODS AND SYSTEMS FOR ARBITRATING ACCESS TO A DISK CONTROLLER MEMORY




(57) Abrégé/Abstract:
The present invention provides a method and system for arbitrating access to a shared disk controller resource. In one embodiment, an access cycle with a first cycle duration is provided. A first amount of time is allocated to a first accessing unit. The

CA 2364625 C 2012/08/07

(11)(21) 2 364 625

(13) C

(57) Abrégé(suite)/Abstract(continued):
said shared controller resource is accessed using the first unit for a first duration, where the first duration being no greater than the first amount of time. A second amount of time and an unused amount of the first amount of time is allocated to a second accessing unit.

**PCT**

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

**(51) International Patent Classification 7 :**
**G06F 3/06**

**A1**

**(11) International Publication Number:** **WO 00/57267**

**(43) International Publication Date:** 28 September 2000 (28.09.00)

**(21) International Application Number:** PCT/US00/07780

**(22) International Filing Date:** 23 March 2000 (23.03.00)

**(30) Priority Data:**
09/275,629 24 March 1999 (24.03.99) US

**(71) Applicant** *(for all designated States except US)*: QLOGIC CORPORATION [US/US]; 26600 Laguna Hills Drive, Aliso Viejo, CA 92656 (US).

**(72) Inventors; and**
**(75) Inventors/Applicants** *(for US only)*: KRANTZ, Leon, A. [US/US]; 27432 Valderas, Mission Viejo, CA 92691 (US). CAMPBELL, Frank, W. [AU/US]; 3535 Conquista Avenue, Long Beach, CA 90808 (US).

**(74) Agent:** NATAUPSKY, Steven, J.; Knobbe, Martens, Olson & Bear, LLP, 16th floor, 620 Newport Center Drive, Newport Beach, CA 92660–8016 (US).

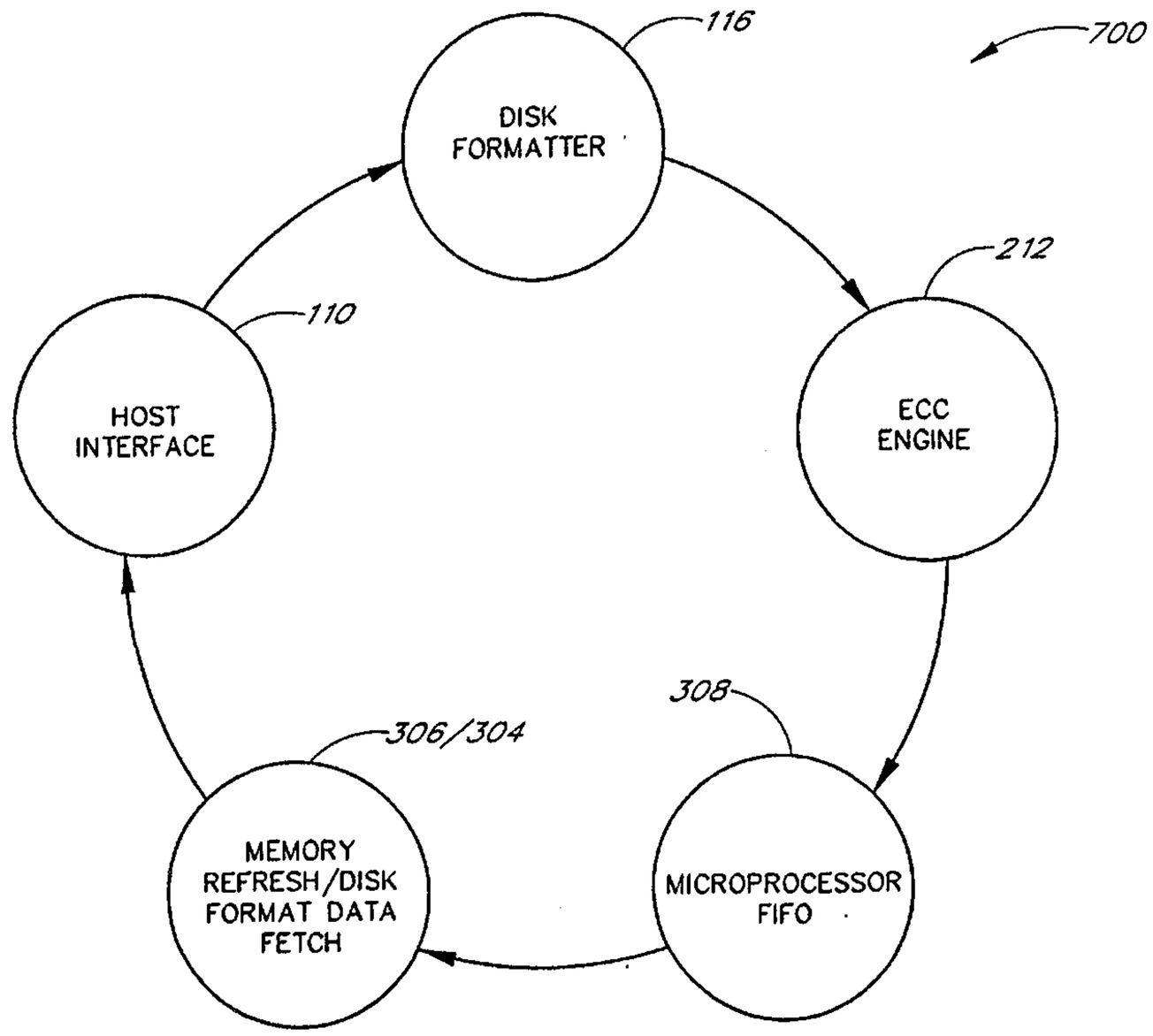
**(81) Designated States:** AE, AG, AL, AM, AT, AT (Utility model), AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, CZ (Utility model), DE, DE (Utility model), DK, DK (Utility model), DM, DZ, EE, EE (Utility model), ES, FI, FI (Utility model), GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KR (Utility model), 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, SK (Utility model), SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, 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**
*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.*

**(54) Title:** ARBITRATION METHODS AND SYSTEMS FOR ARBITRATING ACCESS TO A DISK CONTROLLER MEMORY

**(57) Abstract**

The present invention provides a method and system for arbitrating access to a shared disk controller resource. In one embodiment, an access cycle with a first cycle duration is provided. A first amount of time is allocated to a first accessing unit. The said shared controller resource is accessed using the first unit for a first duration, where the first duration being no greater than the first amount of time. A second amount of time and an unused amount of the first amount of time is allocated to a second accessing unit.



*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 | | | **TR** | Turkey |
| **BG** | Bulgaria | **HU** | Hungary | **ML** | Mali | **TT** | Trinidad and Tobago |
| **BJ** | Benin | **IE** | Ireland | **MN** | Mongolia | **UA** | Ukraine |
| **BR** | Brazil | **IL** | Israel | **MR** | Mauritania | **UG** | Uganda |
| **BY** | Belarus | **IS** | Iceland | **MW** | Malawi | **US** | United States of America |
| **CA** | Canada | **IT** | Italy | **MX** | Mexico | **UZ** | Uzbekistan |
| **CF** | Central African Republic | **JP** | Japan | **NE** | Niger | **VN** | Viet Nam |
| **CG** | Congo | **KE** | Kenya | **NL** | Netherlands | **YU** | Yugoslavia |
| **CH** | Switzerland | **KG** | Kyrgyzstan | **NO** | Norway | **ZW** | Zimbabwe |
| **CI** | Côte d'Ivoire | **KP** | Democratic People's Republic of Korea | **NZ** | New Zealand | | |
| **CM** | Cameroon | | | **PL** | Poland | | |
| **CN** | China | **KR** | Republic of Korea | **PT** | Portugal | | |
| **CU** | Cuba | **KZ** | Kazakstan | **RO** | Romania | | |
| **CZ** | Czech Republic | **LC** | Saint Lucia | **RU** | Russian Federation | | |
| **DE** | Germany | **LI** | Liechtenstein | **SD** | Sudan | | |
| **DK** | Denmark | **LK** | Sri Lanka | **SE** | Sweden | | |
| **EE** | Estonia | **LR** | Liberia | **SG** | Singapore | | |

# ARBITRATION METHODS AND SYSTEMS FOR ARBITRATING ACCESS TO A DISK CONTROLLER MEMORY

## Background of the Invention

### Field of the Invention

The present invention relates to mass data storage device controllers. More particularly, the invention relates to methods and systems for arbitrating access to a disk controller memory.

### Description of the Related Art

Mass data storage is a critical component of computer systems. Typically, mass storage units consist of nonvolatile memory such as a hard disk, an optical drive or other type of storage device. Computer systems typically also contain volatile system memory, such as random access memory (RAM) circuits. Most personal computers today contain less than 256 megabytes of RAM while the same computers often contain hard disks with up to 10 gigabytes or more of capacity.

The mass storage unit of a computer is used to store programs and data on a nonvolatile basis. Thus, the mass storage device retains data even when the computer and the mass storage device are powered down. Volatile system memory, by contrast, serves as a temporary repository of program code and data. Typically, volatile system memory can be accessed much more quickly than a mass storage device. When a microprocessor requires access to programs or data that are not in system memory, the microprocessor first copies the data from the larger, but slower mass storage device to the system memory. The processor may then quickly access the data from the system memory. Further, under certain conditions, if new or modified data is stored in system memory, the microprocessor writes the data to the mass storage device.

A mass storage unit typically consists of a hard disk and a hard disk controller (HDC). The controller operates the hard disk, formats data as it is written to the disk, checks data as it is read from the disk, communicates the data to and from a host system, and buffers the data so as to compensate for latency and difference in data transfer rates between the hard disk and the host computer system.

Typically, the HDC includes a buffer memory which may be used to buffer data between the disk and the host. Because several functional units, such as the disk, the host, and the local controller, need to interact with the buffer memory, access to the buffer memory must be coordinated in some manner. Thus, the HDC may utilize an access arbiter to negotiate access to the buffer memory among various devices requesting access. However, conventional systems typically utilize arbiters that inefficiently provide fixed amounts of access times to accessing units each access cycle. Thus, conventional systems fail to dynamically accommodate changing utilizations of the buffer memory by the accessing units.

## Summary of the Invention

The present invention discloses a method and system for arbitrating access to a shared resource of a mass storage device controller. In one embodiment of the invention, the controller is a hard disk controller and the shared resource is a local buffer memory of the hard disk controller. Several units or circuits within the controller each

access the buffer memory to store or retrieve information. These units may include a controller microprocessor, a host interface unit, a disk formatter unit, a buffer memory refresh unit, a disk format data fetch unit, and an error correction unit. In arbitrating access to the buffer memory, the various access requirements of the units are taken into account. For example, in one embodiment, the disk controller has the most time critical, predictable and periodic access requirements, the other units have less critical and less predictable access requirements, and the host interface unit has the least predictable and least time critical access requirements.

One embodiment of the invention operates in a cycle during which each unit is offered a continuous access duration. The disk formatter, having the most critical access requirements, is offered periodic access such that the time delay during which it does not have access does not exceed a specified time period.

At the high level, the access cycle is bifurcated into two parts. A first part represents the time during which the disk formatter accesses the buffer memory. The second part of the access cycle represents the time during which the remaining units, that is, the units other than the disk formatter, access the buffer memory. However, while the remaining units access the buffer memory, the disk formatter cannot access the buffer memory. In one embodiment, to ensure that the disk formatter is guaranteed periodic access within a predetermined amount of time, the combined access time of the remaining units is limited to a predetermined amount. Similarly, in order to ensure that the remaining units receive access, the access time of the disk formatter is also limited to a predetermined amount. As each of the two parts of the access cycle has a maximum duration, the access cycle itself has a maximum allowed duration.

At the next level, the part of the access cycle allocated to the remaining units is again divided up. Each of the remaining units is allocated a maximum amount of time within the time allocated to the remaining, non-disk formatter units. If any of the remaining units do not use their maximum allocation, the unused time is offered to the last of the remaining units. If the last unit does not use all of the time available to it, the arbitration cycle terminates early. Thus, the time before which the disk formatter is again offered access to the buffer memory ends up being shorter than the specified maximum duration.

In one embodiment, during an access cycle, time is deducted from the total access time allocated to the remaining units as each of the remaining units sequentially utilizes a portion of the total allocated time. Whatever remains of the allocated time after all but the last unit has been offered access, is allocated to the last unit.

In accordance with an aspect of the present invention, there is provided a method for providing access to a buffer memory of a hard disk controller comprising the acts of:

allocating a first amount of time to an error correction unit;

allocating a second amount of time to a host interface unit;

accessing said buffer memory using said error correction unit for a first duration, said first duration being no greater than said first amount of time;

adjusting said second amount of time by an amount related to the difference between said first amount of time and said first duration; and

accessing said buffer memory using said host interface unit for a second duration, said second duration being no greater than said second amount of time.

In accordance with another aspect of the present invention, there is provided a method for providing access to a shared controller resource of a mass storage device controller comprising the acts of:

providing a first access cycle during which access to said shared controller resource is arbitrated;

allocating a first portion of said first access cycle as a first continuous block of access time to a first accessing unit;

utilizing an amount of said first continuous block of access time using said first accessing unit;

allocating a second portion of said first access cycle and an unused amount of said first continuous block of access time as a second continuous block of access time to a second accessing unit; and

utilizing an amount of said second continuous block of access time using said second accessing unit.

In accordance with another aspect of the present invention, there is provided a method of varying a duration of an access cycle of a mass storage device controller shared resource, wherein the access of multiple units to the shared controller resource is arbitrated, said method comprising the acts of:

providing an access cycle, said access cycle having a cycle duration;

allocating a first amount of time of said cycle duration to a first accessing unit;

accessing said shared controller resource using said first unit for a first duration, said first duration being no greater than said first amount of time; and

allocating a second amount of time and an unused amount of said first amount of time to a second accessing unit, wherein said unused amount of said first amount of time is based on a difference between said first amount of time and said first duration.

In accordance with another aspect of the present invention, there is provided a method for providing access to a shared memory of a disk controller, said method comprising the acts of:

allocating an amount of time as an available memory access time to each of at least two units; and

accessing said shared controller memory using said at least two units, wherein unused time allocated to a first of said at least two units is reallocated to a second of said at least two units.

In accordance with another aspect of the present invention, there is provided a method for providing access to a shared controller resource of a mass storage device controller comprising the acts of:

allocating a first amount of time to a first accessing unit;

allocating a second amount of time to a second accessing unit;

accessing said shared controller resource using said first unit for a first duration, said first duration being no greater than said first amount of time; and

adjusting said second amount of time by an amount related to the difference between said first amount of time and said first duration; and

accessing said shared controller resource using said second unit for a second duration, said second duration being no greater than said adjusted second amount of time.

In accordance with another aspect of the present invention, there is provided a mass storage device controller circuit, including a memory shared by multiple circuits, said controller circuit comprising:

an error correction circuit used to correct data read from a mass storage device, said error correct circuit associated with a first amount of time allocated to access said shared memory;

a host system interface circuit used to exchange data with a host system, said host system interface circuit associated with a second amount of time allocated to access said shared memory;

a storage device formatter circuit used to interface to said mass storage device, said storage device formatter circuit associated with a third amount of time allocated to access said shared memory; and

an arbiter circuit used to arbitrate access to said shared memory by at least said host system interface circuit, said error correction circuit, and said storage device formatter circuit, wherein said arbitration circuit is configured to grant said error correction circuit shared memory access for a first duration, said first duration being no greater than said first amount of time, and to adjust said second amount of time by an amount related to the difference between said first amount of time and said first duration granting buffer shared memory access to said disk formatter unit,

wherein said arbiter circuit grants said storage device formatter circuit shared memory access for a third duration, said third duration being no greater than said third amount of time.

In accordance with another aspect of the present invention, there is provided a computer system comprising:

a host system;

a host system interface circuit coupled to said host system, said host system interface circuit used to exchange data with said host system;

an error correction circuit used to correct erroneous data read from a mass storage device;

a storage device formatter circuit used to interface to said mass storage device; and

an arbiter circuit used to arbitrate access to a buffer memory by at least said host system interface circuit, said error correction circuit, and said storage device formatter circuit, wherein said arbiter circuit is configured to allocate a first amount of time to said error correction circuit, to allocate a second amount of time to said host interface circuit, and to allocate a third amount of time to said

storage device formatter circuit, and said arbiter circuit is further configured to provide said error correction circuit access for a first duration, said first duration being no greater than said first amount of time, to adjust said second amount of time by an amount related to the difference between said first amount of time and said first duration, and to provide said storage device formatter circuit access for a third duration no greater than said third amount of time.

In accordance with another aspect of the present invention, there is provided a mass storage device controller comprising:

a means for buffering data;

a means for exchanging data with a host system, said means for exchanging data with a host system coupled to said means for buffering data;

a means for processing data coupled to said means for buffering data;

a means for exchanging data with a mass storage device, said means for exchanging data with said mass storage device coupled to said means for buffering data; and

a means for arbitrating access to said means for buffering data by at least said means for exchanging data with a host system, said means for processing data, and said means for exchanging data with a mass storage device, wherein said means for arbitrating access is configured to allocate a first amount of time to said means for processing data, to allocate a second amount of time to said means for exchanging data with a host system, and to allocate a third amount of time to said means for exchanging data with a mass storage device, wherein said means for arbitrating is further configured to provide said means for processing data access for a first duration, said first duration being no greater than said first amount of time, to adjust said second amount of time by an amount related to the difference between said first amount of time and said first duration, and to provide said means for exchanging data with a mass storage device access for a third duration no greater than said third amount of time.

In accordance with another aspect of the present invention, there is provided a method for providing access to a shared controller resource of a mass storage device controller comprising the acts of:

allocating a first amount of time as a first available access time to a first accessing unit;

allocating a second amount of time as a second available access time to a second accessing unit, said second available access time including at least a portion of said first available access time;

accessing said shared controller resource using said first unit for a first duration, said first duration being no greater than said first available access time;

decreasing said second available access time by said first duration; and

accessing said shared controller resource using said second unit for a second duration, said second duration being no greater than said decreased second available access time.

In accordance with another aspect of the present invention, there is provided a disk controller circuit, including a memory accessed by a plurality of disk controller circuits, said disk controller circuit comprising:

a first controller circuit with a corresponding first allocated access duration;

a second controller circuit with a corresponding second allocated access duration; and

an arbiter circuit which measures a quantity associated with an actual access duration by said first controller circuit and adjusts said second allocated access duration by an amount related to a difference between said first allocated access duration and said actual access duration.

## Brief Description of the Drawings

Referring now to the drawings in which like reference numbers represent corresponding components throughout:

Figure 1 illustrates a block diagram of one embodiment of a computer system adapted to implement one embodiment of the present invention;

Figure 2 illustrates one embodiment of a disk controller circuit;

Figure 3 illustrates one embodiment of the bus architecture of the disk controller;

Figure 4 illustrates one embodiment of the disk controller arbitration architecture;

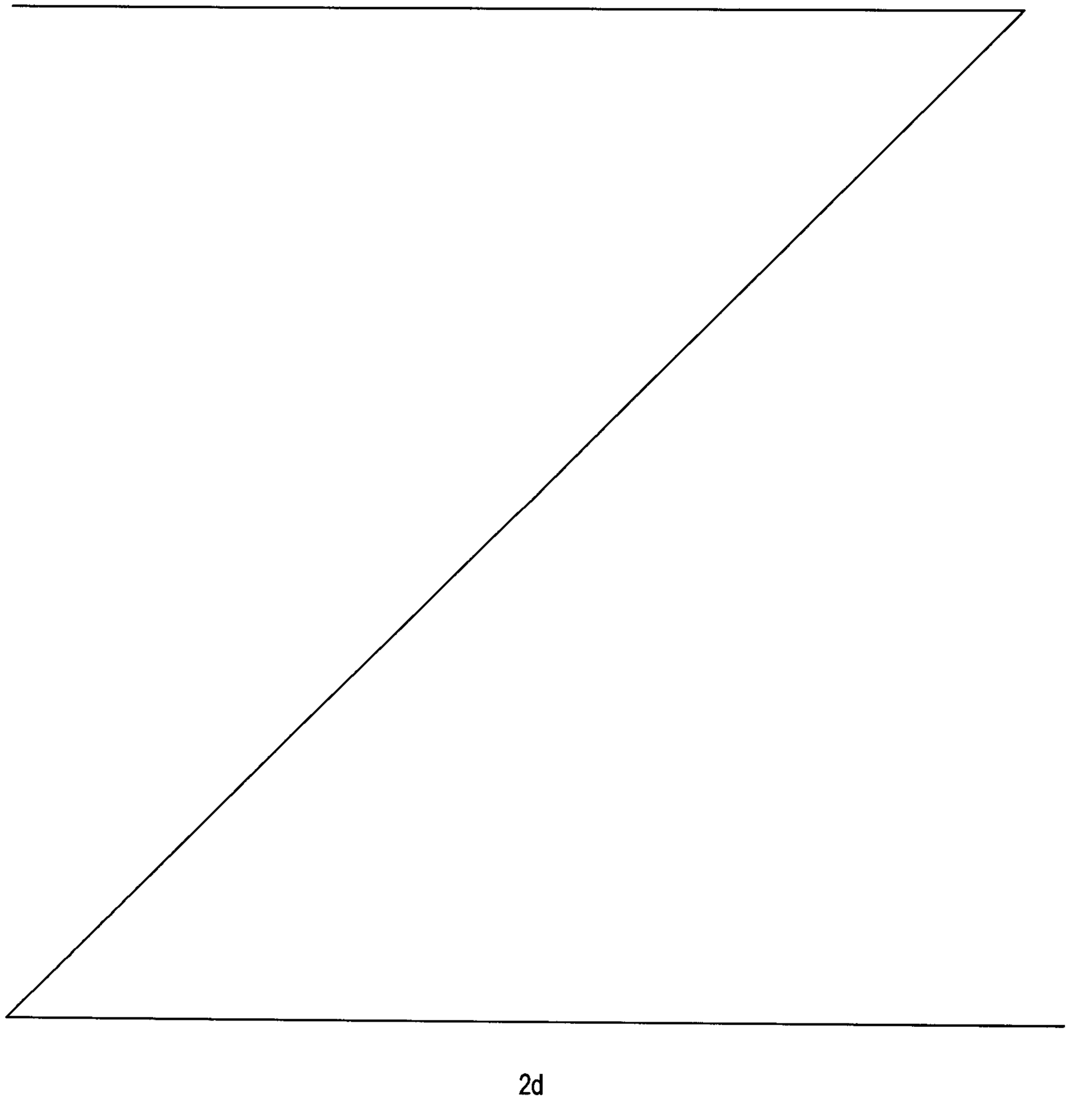


Figure 5 illustrates one embodiment of a typical disk write sequence and data format;

Figure 6 illustrates one embodiment of a typical disk read sequence and data format;

Figure 7 illustrates one embodiment of an arbitration sequence;

Figure 8A-8E illustrate a flow chart of one embodiment of an arbitration process; and

Figure 9 shows a time line of the arbitration limits and some exemplary access timing diagrams.

## Detailed Description of the Drawings

In the following description, reference is made to the accompanying drawings, which form a part hereof, and in which is shown by way of illustration a specific embodiment in which the invention may be practiced. It is to be understood that other embodiments may be utilized and structural changes may be made without departing from the scope of the present invention. Where possible, the same reference numbers will be used throughout the drawings to refer to the same or like components. Numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be obvious to one skilled in the art that the present invention may be practiced without the specific details or with certain alternative equivalent devices and methods to those described herein. In other instances, well-known methods, procedures, components and devices have not been described in detail so as not to unnecessarily obscure aspects of the present invention.

The detailed description which follows is organized into the following sections: Architectural Overview of One Embodiment of a System Appropriate for Use with One Embodiment of the Present Invention, Detailed Description of One Embodiment of the Arbitration Process, Exemplary Timing Scenarios, and Exemplary Arbitration Limit Calculations.

## Architectural Overview of One Embodiment of a System Appropriate for Use with One Embodiment of the Present Invention

Figure 1 illustrates a block diagram of one embodiment of a system 100 in which a hard disk controller (HDC) 122 is typically used. An HDC core 102 is connected via a host bus interface 110 to a host system 108 using a host system bus 112. The host system 108 may consist of one or more host system microprocessors, system RAM, a monitor, a keyboard, and other devices.

The host system 108 exchanges data to be stored on or retrieved from a hard disk 114 through the host system bus 112. Instructions from the host system 108 to the HDC core 102 regarding, for example, what data to retrieve or where to store data are also sent over the host system bus 112.

In addition, a controller microprocessor 104 is connected to the HDC core 102 through a microprocessor interface 106. The controller microprocessor 104, which, in one embodiment, is distinct from the host system microprocessor, receives and executes instructions from the host system 108 and manages the HDC core 102. The hard disk 114 is connected to the HDC core 102 through a disk formatter 116. As discussed in greater detail below, the disk formatter 116 controls the operation of the hard disk 114. In the illustrated embodiment, the hard disk 114 consists of a magnetic disk device and associated circuitry. A controller buffer memory 118 is connected to the HDC

core 102 through a buffer interface 120. In one embodiment, the buffer memory 118 is configured using random access memory (RAM). The RAM may be dynamic or static.

Figure 2 illustrates one embodiment of the HDC core 102 and associated ports 204, 208, 210, 216. The bus interface 202 sends and receives information from the host system 108 through data, control, and interrupt lines of the host bus port 204. The microprocessor interface logic 206 connects the controller microprocessor 104 to the HDC core 102 through timing, address, and data lines of the controller microprocessor port 208. The microprocessor bus 220 allows the controller microprocessor 104 to communicate with and control the various units in the HDC 102 core.

As, illustrated in Figure 2, the disk formatter 116 is connected to the disk 114 through the data and control lines of the disk port 210. A Reed Soloman error correction code (ECC) engine 212 is connected to the disk formatter 116 via a bus. The ECC engine 212 performs error correction on disk reads, as well as error correction code generation on disk writes. A buffer controller 214 is connected to the buffer memory 118 through the address, timing, and data lines of the buffer port 216. In one embodiment, the buffer controller 214 includes an arbiter (not shown) which controls access to the buffer memory 118 by the disk formatter 116, the ECC engine 212, the host bus interface 202, and the microprocessor interface 206, in addition to other units discussed below. The arbiter is later discussed in greater detail. The various units that access the buffer memory 118 do so through the data bus 218. The data bus 218 is coupled to the buffer memory 118 via the buffer port 216.

Figure 3 depicts one embodiment of the HDC's bus architecture. The units which access the buffer memory 118 do so through respective "first in first out" (FIFO) buffers 308, 312, 314, 316, 318 and 320. These FIFO buffers may vary in size. In one embodiment, the FIFO buffers each have between 64 to 96 bytes of storage. The controller microprocessor 104 accesses the buffer memory 118 through the microprocessor FIFO 308. The buffer memory 118 serves as the main memory, storing instructions and data, for the controller microprocessor 104.

A disk format data fetch unit 304 operates to fetch disk format data from the buffer memory 118. The disk formatter 116 uses this data to locate and organize host system data on the hard disk 114. The memory refresh unit 306 operates to refresh the buffer memory 118 at required intervals when the buffer memory uses dynamic RAM. As discussed below, in one embodiment, both the disk format data fetch unit 304 and the memory refresh unit 306 share one arbitration slot for the purposes of buffer memory access arbitration. Thus, only one of the disk format data fetch unit 304 and the memory refresh unit 306 may access the buffer memory 118 in a single access cycle. However, in another embodiment, the memory refresh unit 306 and disk format data fetch unit 304 may each have their own arbitration slot. The disk format data fetch unit 304 and the memory refresh unit 306 access the buffer memory 118 through their respective FIFO buffers 314 and 316.

As illustrated in Figure 3, the ECC engine 212 is connected to the buffer memory 118 through a FIFO 318. The ECC engine FIFO 318 is used to correct data read from the hard disk 114 while the data is in the buffer memory 118. The buffer controller 214 effectuates the correction through a read/modify/write process.

The host bus interface 110 is connected to the buffer memory 118 through its FIFO 320. The host bus interface 110 channels data and instructions between the host system 108 and the buffer memory 118.

Figure 4 illustrates one embodiment of the connectivity between the disk formatter 116, the disk format data fetch unit 304, the memory refresh unit 306, the ECC engine 212, the microprocessor FIFO 308, the host bus interface 110 and an arbiter 402. The arbiter 402 arbitrates between the various units accessing the buffer memory. The arbiter 402 grants access to requesting units during a repeating access cycle. An access cycle is a period of time during which access is offered to each unit, or a group of units sharing an access slot, that access the buffer memory. In one embodiment, each unit has a request line, titled "REQ," for requesting access to the buffer memory 118, as well as an acknowledge line, titled "ACK," by way of which the arbiter 402 grants a requested access.

Connected to the arbiter 402 is a set of arbitration limit registers 410-418, which hold respective units' arbitration limits. In one embodiment a given arbitration limit is specified as the number of HDC clock cycles per access cycle available to a unit during which it can access the buffer memory 118. The arbitration limit registers include: the disk formatter arbitration limit register 410, the shared disk format data fetch unit and memory refresh unit arbitration limit register 412, the ECC engine arbitration limit register 414, the microprocessor FIFO arbitration limit register 416, and the host bus interface (global) arbitration limit register 418. Note that the disk format data fetch unit 306 and the memory refresh unit 306 share the same arbitration limit and use the same arbitration limit register 412. The host bus interface arbitration limit register 418 is also called the global arbitration limit register for reasons that will be discussed below. The arbitration limit registers 410-418 store time limits defined in HDC clock cycles for each unit's maximum specified access duration during an access cycle. In one embodiment, the access duration is for a continuous time period. As further described below, also connected to the arbiter 402 are an arbitration limit counter 406 and a global arbitration limit counter 408, used to keep track of the access durations of the units by loading and decrementing arbitration limit values. The global arbitration limit counter 408 tracks the available access time of the host bus interface and the arbitration limit counter 406 tracks the available access time for the remaining units.

Figure 5 illustrates an exemplary disk write sequence 500 as the write data passes through various units of the HDC 102. Data 502 from the host system 108 arrives at the host bus interface 110 and is passed through the host bus interface FIFO 320 to the buffer memory 118. The FIFO 320 buffers the data 502 as needed while waiting for access to the buffer memory 118. In one embodiment, data is read from or stored on the disk 114 in units no smaller than a sector, which is typically 512 bytes. If a quantity of data less than a sector needs to be accessed, a whole sector is nevertheless accessed. After at least a complete sector of data is stored in the buffer memory 118, the data 502 is then directed to the disk formatter 116 via the disk formatter FIFO 312 along with a cyclic redundancy check (CRC) tag. The CRC is a code generated from the data that can be used to confirm the correctness of the data. The FIFO 312 buffers the data 502 as it arrives from the buffer memory 118. While the data 502 and CRC 508 are en route to the disk formatter 116, the data 502 and CRC 508 are also passed through the ECC engine 212, which generates an error correction code (ECC) 510. The ECC 510 is passed to the disk formatter 116 to be

appended to the data 502 and the CRC 508, all of which is then written to the hard disk 114 by the disk formatter 116.

Figure 6 illustrates an exemplary disk read sequence 600 as the data passes through the units of the HDC 102. ECC techniques are used to compensate for the possibility that a sector's data 604, CRC 606 and ECC 602 might become corrupted during the process of being written and read from the hard disk 114. As described below, the ECC engine 212 performs the task of correcting the uncorrected data 604 and/or CRC 606 after it has been retrieved from the hard disk 114. As the uncorrected data 604, CRC 606 and ECC 602 are read from the hard disk 114, the disk formatter 116 sends the uncorrected data 604, CRC 606 and ECC 602 to the ECC engine 212. The disk formatter 116 also strips off the ECC 602 and passes the uncorrected data 604 and CRC 606 to the disk formatter FIFO 312 and then on to the buffer memory 118. The disk formatter FIFO 312 absorbs any latency when waiting to access the buffer memory 118. While the uncorrected data 604 and CRC 606 are stored in the buffer memory 118, the ECC engine 212 executes an algorithm that creates error correction information 612 from the uncorrected data 604, CRC 600, and ECC 602. The ECC engine 212 passes this error correction information 612 on to the ECC FIFO 318. The buffer controller 214 then uses the error correction information 612 in the ECC FIFO 318 to attempt to correct any errors in the uncorrected data 604 and the CRC 606 residing in the buffer memory 118. The buffer controller 214 corrects the data 604 and the CRC 606 by reading the uncorrected data 604 and/or CRC 606 from the buffer memory 118, appropriately modifying the uncorrected data 604 and/or CRC 606 using the information 614 in the ECC FIFO 318 and then writing the corrected data 616 and/or CRC 618 back to the buffer memory 118. The corrected data 616 and CRC 618 are then passed to the host bus interface FIFO 320. The CRC 618 is then checked. The CRC check determines whether the data correction operation was successful. If the data 616 is correct, the data 616 is passed to the host bus interface 110 and on to the host system 108. If the data 616 is not correct, an error signal is generated and directed to the host system.

As discussed above, several units within the HDC 102 access the buffer memory 118 during either a typical read or write of disk data. In addition to the accesses described during the read and write sequences, the controller microprocessor 104 also accesses the buffer memory 118 since the buffer memory serves as the microprocessor's main memory. Furthermore, upon performing a disk write or read, the disk formatter 116 utilizes disk format information retrieved by the disk format data fetch unit 304 and stored in the buffer memory 118. The buffer memory's dynamic RAM 118 also needs to be refreshed periodically and thus the buffer memory refresh unit 306 accesses the buffer memory as well.

## Detailed Description of One Embodiment of the Arbitration Process

One embodiment of the arbitration method and apparatus ensures periodic access by all units requesting access and dynamically provides for an efficient allocation of available access time among units. The arbiter 402 functions through the use of settable arbitration limits. In one embodiment, the arbitration limits correspond to the numbers of clock cycles representing the maximum access time for a corresponding unit. The limits may be

programmed in non-volatile memory such as ROM or EEPROM. Alternatively, the arbitration limits may be determined and set on the fly by the host system or the controller microprocessor.

Figure 7 illustrates one embodiment of an access cycle 700. During an access cycle, the arbiter 402 checks each unit for access requests in a predetermined, repeating order. If a unit has not raised its REQ line to the arbiter 402, the arbiter 402 then checks the next unit's REQ line. The arbiter 402 continues around the access cycle until it finds a unit requesting buffer memory access. Note that memory refresh unit 306 and disk format data fetch unit 304 compete for the same access cycle slot. In a given access cycle, the arbiter may only grant access to one of the memory refresh unit 306 or the disk format data fetch unit 304. If both units 304/306 have requested access in the same cycle, the arbiter 402 gives priority to the memory refresh unit 306 as the buffer memory needs to be refreshed to avoid losing its contents; the disk format data fetch unit 304 then waits until the next access cycle to gain access.

Figure 8 shows a detailed flowchart 800 of one embodiment of a single access cycle of the arbitration process. In one embodiment, the arbitration process is controlled using an arbiter state machine. Because the arbitration process runs in a cycle, the choice of a start location is not critical. For the purposes of illustration, however, a start state 802 is shown. The first portion of the arbitration process 804-814 is directed to disk formatter access. At state 804, the arbiter 402 checks whether the disk formatter 116 is requesting buffer memory access by checking the disk formatter REQ line. If the disk formatter 116 is requesting access, the arbiter 402 grants access, at state 806, by raising the disk formatter's ACK line. Proceeding to state 808, the arbiter 402 reads the disk formatter arbitration limit from the corresponding arbitration limit register 410 and loads the limit into the arbitration limit counter 406. The arbitration limit counter 406 is decremented (counts down by one) during each HDC clock cycle. Next, at states 810, 812, the arbiter 402 repeatedly checks whether the arbitration limit counter 406 has reached zero and whether the disk formatter's request line REQ is still active. If the arbitration limit counter 406 has reached zero or if the disk formatter's request line REQ is inactive, at stage 814 the arbiter 402 deasserts the disk formatter's acknowledge ACK line. If, at state 804 mentioned above, the disk formatter 116 is not requesting access, the arbiter proceeds directly from state 804 to state 816.

Before proceeding with the rest of the arbitration process, at state 816 the arbiter 402 loads the global arbitration limit from the global arbitration limit register 418 into the global arbitration limit counter 408. The global arbitration limit counter 408 is decremented during each clock cycle of the remainder of the access cycle. Thus, the global arbitration limit is decremented while the ECC engine 212, the microprocessor FIFO 308, the disk format data fetch/memory refresh units 304/306 and the bus interface unit 110 are accessing the buffer memory 118. Once the global arbitration limit counter 408 reaches zero, a new access cycle begins and access is again offered to the disk formatter 116.

The global arbitration limit serves two purposes. First, the global arbitration limit limits the time available to all units other than the disk formatter 116. Second, the global arbitration limit serves, in effect, as the arbitration limit for the host interface unit 110. Each of the ECC engine 212, the microprocessor FIFO 308, and the disk format data fetch/memory refresh units 304/306 has its own arbitration limit. While each of these units are accessing the

buffer memory 118, the arbitration limit counter 406 counts down from the corresponding arbitration limits. Further, while each of these units are accessing the buffer memory 118, the global arbitration limit counter 408 counts down from the global arbitration limit to determine the time that will eventually be offered to the host interface unit 110. Once the ECC engine 212, the microprocessor FIFO 308, and the disk format data fetch/memory refresh units 304/306 have completed their accesses, the host interface unit 110 is given access for the time remaining in the global arbitration limit counter 408.

Once the global arbitration limit counter 408 has been loaded, the arbiter 402 proceeds to state 818 to determine if the ECC engine 212 is requesting access by checking the ECC engine REQ line. If the ECC engine 212 is requesting access, the arbiter proceeds to state 820 and grants access by asserting the ECC engine's ACK line. At this point, the arbiter 402 reads the ECC engine arbitration limit from the corresponding arbitration limit register 412, and proceeding to state 822, sets the arbitration limit counter 406. The arbitration limit counter 406 is decremented each clock cycle during the ECC engine access time period. Note that the global arbitration limit counter 408 is also decremented during this time. Next, the arbiter 402 at states 824. 826 repeatedly checks whether the arbitration limit counter 406 has reached zero and whether the ECC engine's request line REQ is still raised. If the arbitration limit counter 406 has reached zero or if the ECC engine's request line REQ is inactive, the arbiter 402 deasserts the ECC engine's ACK line at state 828. If, at state 818, the ECC engine 212 is not requesting access, the arbiter 402 proceeds directly to state 830.

Once the arbiter 402 has completed the ECC engine portion of the access cycle, the arbiter 402 proceeds to offer access to the microprocessor FIFO 308 at states 830-840 in a manner similar to that described above with respect to the ECC engine access. As with the ECC engine 212, while access is granted to the microprocessor FIFO 308, both the arbitration limit counter 406 and global arbitration limit counter 408 are decremented each clock cycle. Once the arbiter 402 has completed the microprocessor FIFO portion of the access cycle, the arbiter 402 proceeds to state 842.

In the present embodiment, the arbiter 402 grants access to only one of the memory refresh unit 306 and the disk format data fetch unit 304 in a single access cycle. Thus, the flowchart branches at state 842 to accommodate this feature. A buffer memory refresh unit request takes priority over a disk format data fetch unit request. Therefore, the arbiter 402 first determines whether the refresh unit 306 is requesting access. If the refresh unit 306 is requesting access, the arbiter 402 follows a sequence 844-852 similar to those described above for the previous units. Of course, in this case the arbitration limit counter 406 is loaded with the refresh/data fetch arbitration limit. If the refresh unit 306 is not requesting access, the arbiter 402 proceeds to state 854 and determines whether the disk format data fetch unit 304 is requesting access. If so, then the arbiter 402 follows a sequence 856-864 similar to the sequence for the refresh unit 306. If neither the disk format data fetch unit 304 nor the refresh unit 306 request access, the arbiter 402 proceeds directly from state 854 to state 866. At this point the arbiter 402 offers access to the host interface 110.

In the present embodiment, the access duration of the host interface 110 is limited only by the global arbitration limit. As previously discussed, the global arbitration limit counter 406 is reset after access is offered to the disk formatter 116 and before access is offered to the ECC engine 212 and is then continually decremented while the units previous to the host interface 110 access the buffer memory 118. The host interface 110 is allowed access for the remainder of the global arbitration limit not used by the previous units. Note that the amount of time during which the disk formatter 116 accesses the buffer memory 28 does not affect the global arbitration limit counter 408 or the amount of time allocated to the host interface 110. This is because the global arbitration limit counter 408 is set only after the disk formatter 116 has had its turn during the access cycle. Thus, the host interface is flexibly offered an access time ranging from a predetermined minimum to a maximum as large as the global arbitration limit in a continuous block of time.

To allocate access to the host interface 110, at state 866, the arbiter first checks whether the host interface 110 is requesting access. If so, at state 868, the arbiter asserts host interface ACK line, thereby granting access to the buffer memory 118. The global arbitration limit counter 408 continues to be decremented while the host interface unit 110 is granted access. In the present embodiment, the global arbitration limit is sufficiently large to accommodate the arbitration limits of the previous units and still have at least some remaining time for the host interface unit. Next, at states 870, 872 the arbiter 402 repeatedly checks to see whether the global arbitration limit counter 408 has reached zero and whether the host interface unit 110 is still requesting access. If the global arbitration limit counter 408 has reached zero, or if the host interface unit 110 drops its REQ line at state 874, the arbiter 402 deasserts the host interface's ACK line. If, at state 866, the host interface 110 is not requesting access, the arbiter 402 proceeds from state 866 to state 876. At this point the arbiter 402 has completed one access cycle and returns to the start state 802 of the flowchart 800.

In another embodiment of the present invention, the host interface unit's arbitration limit can be determined by an additive rather than a subtractive method. In this case, the global arbitration limit counter 408 would be initialized with a minimum arbitration limit for the host interface 110. Upon the release of access by any of the ECC engine 212, the microprocessor FIFO 308, the memory refresh 306 or the disk format data fetch unit 304, the value remaining in the arbitration limit counter 406 would be added to the global arbitration limit counter 408. The global arbitration limit counter 408 would then be decremented only while the host interface 110 is accessing the buffer memory 118. This process would yield the same result as the technique already described in detail above. In another embodiment, unused access times of earlier accessing units could be allocated to any subsequent unit. In still another embodiment, unused access times of earlier accessing units could be allocated to a particular subsequent unit other than the last accessing unit.

Exemplary Timing Scenarios

Figure 9 shows several exemplary arbitration limits and associated timing diagrams illustrating the arbiter operation. As mentioned above in the illustrated embodiment, an arbitration limit denotes the number of HDC clock cycles of each access cycle available to a corresponding unit during which the unit can access the buffer memory 118.

With regard to the illustrated time lines 902-918, time begins towards the left side of the figure. The first time line 902 illustrates that at a top level, the arbitration process can be divided into two parts. The first part consists of the time allocated to the disk formatter 116, the disk arbitration limit. The second part consists of the time allocated to all of the other units, the global arbitration limit. The second time line 904 illustrates that the time available to the host interface 110, represented by the dynamic host interface arbitration limit, is the time represented by the global arbitration limit, diminished by the time granted to the ECC 212, microprocessor 308, refresh/data fetch 304/306 units. Time line 906 illustrates that in one embodiment, the combined ECC arbitration limit, the microprocessor arbitration limit, and the refresh/format data fetch arbitration limits are less than the global arbitration limit, thereby ensuring a minimum amount of time, the minimum host interface arbitration limit, is allocated to the host interface unit 110. Thus, even if the ECC 212, microprocessor 308 and the refresh/data fetch 304/306 units use all of their available access time, there will still be time remaining for the host interface 110 to access the buffer memory 118.

Figure 9 illustrates timing diagrams for exemplary buffer memory arbitration sequences 908-918. In each diagram, one complete access cycle is shown, illustrating how the access cycle time can vary. The first exemplary arbitration sequence 908 shows the relative access time allocated to each unit when all units are requesting their maximum access time. In this case, the total access cycle time is at its maximum. In the next sequence 910, the ECC engine 212 deasserts its REQ line and ceases access about halfway through the allocated arbitration time shown in 906. In this example, the microprocessor FIFO 308 does not request access at all, but the host interface 110 requests, and is granted all of the remaining available time. In the next sequence 912, the ECC engine 212, refresh/data fetch units 304/306 and microprocessor 308 do not request buffer access. Thus, the complete global arbitration limit is made available to the host interface 110. In this sequence, both the disk formatter 116 and the host interface 110 use all of the time available to each.

In the next sequence 914, the disk formatter 116 releases its REQ line and only uses about half of its available time. In this sequence, the ECC engine 212, refresh/data fetch units 304/306, and the host interface 110 use all of their available access time. Because the disk formatter 116 relinquishes its access early, the complete access cycle time is less than in the previous sequences. The sequence 916 is similar to the sequence 914, except that the ECC 212, microprocessor 308 and refresh/data fetch 304/306 units do not request access. However, the host interface 110 requests maximum access in this case and uses the time given up by the other units. The last sequence 918 depicts a situation where the disk formatter 116 requests its full access time, the ECC 212, microprocessor 308 or refresh/data fetch 304/306 units do not request access, and the host interface 110 only requests a fraction of its available access time. In this case, because the global arbitration limit is never reached, the total access cycle time is substantially less than the maximum possible access cycle time.

Exemplary Arbitration Limit Calculations

Arbitration limits may be based on several factors including the following: the disk formatter FIFO size, the width of the buffer memory data bus, the hard disk data read/write rate, the clock speed of the HDC 102, the time delay before requested data is read from the buffer memory 118 and the time it takes for the arbiter 402 to perform

its operations, among others. These factors are typically constant and can be determined at the time of design and manufacture of the hard disk unit. The arbitration limits can be stored in non-volatile memory, or alternatively, these factors can be determined dynamically, during system operation.

In one embodiment of the present invention, the global arbitration limit is set to optimize the utilization of the hard disk 114. This is accomplished by ensuring that the disk formatter FIFO 312 is never left empty or full. The global arbitration limit defines the number of clock cycles between the time the disk formatter 116 gives up access to the buffer memory 118 on a previous cycle and the time that data again arrives at the FIFO 312 from the buffer memory 118 on a subsequent cycle. As long as there is data in the FIFO 312 on a disk write and as long as the FIFO 312 is not full on a disk read, the hard disk 114 will be optimally utilized.

Referring back to Figure 5, during a disk write operation, data 502 is transferred from the buffer memory 118 to the disk formatter FIFO 312. When the disk formatter 116 is not accessing the buffer memory 118, the disk formatter FIFO 312 is continually emptied, with data being written to the hard disk 114 at the disk data rate. The disk data rate is the rate at which data can be written to or read from the hard disk 114. In one embodiment, the time it takes to empty a full disk formatter FIFO 312 can thus be defined as:

*Time to empty = (FIFO size)/(disk data rate)*

This time can be multiplied by the clock frequency to yield the time in clock cycles.

*Clock cycles to empty = (clock frequency)*(FIFO size)/(disk data rate)*

By setting the global arbitration limit to approximately the number of clock cycles it takes for the hard disk 114 to empty a full disk formatter FIFO 312, the disk formatter 116 will be able to begin refilling the FIFO 312 just as it becomes empty. Thus, disk utilization is optimized.

In setting the global arbitration limit, some additional considerations may be taken into account. First, the arbiter 402 takes a finite number of clock cycles, called the arbitration delay, in order to grant access to a unit. Second, it takes a finite number of clock cycles, called the setup time, in order to set up the buffer memory 118 to begin transferring data. Third, it takes a finite number of clock cycles, called the read delay, to transfer data from the buffer memory 118 to the disk formatter FIFO 312. Thus, the number of "clock cycles to empty" is decreased by the arbitration delay, the setup time and the read delay, so that data from the buffer memory 118 arrives at the disk formatter FIFO 312 just as the FIFO 312 becomes empty. With these adjustments in one embodiment, the global arbitration limit is calculated as follows:

*Global arbitration limit = (clock frequency)*(FIFO size)/(disk data rate) – (arbitration delay) – (setup delay) – (read delay)*

Although this global arbitration limit is based on the timing of a disk write, the timing of a disk read is analogous and yields a similar result.

The same adjustments made to the global arbitration limit are added, instead of subtracted, to the disk formatter arbitration limit. In essence, the cycles taken from the global arbitration limit are given to the disk

formatter arbitration limit to maximize disk formatter utilization. However, before making the adjustments to the disk formatter arbitration limit, a base arbitration limit is formulated.

In calculating the disk formatter arbitration limit, the fact that the disk formatter FIFO 312 is being both emptied and filled simultaneously needs to be taken into consideration. In the preferred embodiment, when the disk formatter 116 has access to and is transferring data from the buffer memory 118 to the disk formatter FIFO 312, the disk formatter 116 is simultaneously emptying the FIFO 312 to the hard disk 114. When the disk formatter FIFO 312 is being simultaneously filled and emptied, the net rate at which it is being filled is the difference in the rates at which data is being placed in the FIFO 312 and at which data is being removed from the FIFO 312. The rate at which the data is placed in the FIFO 312, called the buffer data rate, is equal to the HDC's bus width times the clock frequency. With this, the fill rate can be expressed as follows:

*Fill rate = (bus width)*(clock frequency) – (disk data rate)*

The disk formatter FIFO size is divided by the fill rate to determine the time to fill the disk formatter FIFO 312. This result is multiplied by the clock frequency to yield the number of clock cycles to fill the disk formatter FIFO 312:

*Clock cycles to fill = (FIFO size)*(clock frequency)/((bus width)*(clock frequency) – (disk data rate))*

As mentioned above in the calculations of the number of cycles to fill the disk formatter FIFO 312, the arbitration, the setup and the delay times are taken into account. As the FIFO 312 is being filled, however, there is a chance that a page boundary in the buffer memory 118 may be crossed. When a page boundary is crossed, another setup delay is incurred, and so a second setup delay is included in the calculation. The resulting disk formatter arbitration limit is thus increased by the additional overhead time of the arbitration delay, twice the setup delay, and the read delay:

*Disk formatter arbitration limit = (FIFO size)*(clock frequency)/((bus width)*(clock frequency) – (disk data rate)) + (arbitration delay) + 2*(setup delay) + (read delay)*

With the disk formatter and global arbitration limits set as described above, the disk formatter 116 and hard disk 114 achieve optimal utilization. The arbitration limits of the other units within the HDC 102 are constrained by the global arbitration limit. The memory refresh/disk format data fetch units 304/306 and the ECC engine 212 typically have predictable buffer utilization characteristics. Thus, arbitration limits can be easily selected in accordance with those characteristics. The microprocessor FIFO's arbitration limit accommodates a reasonable expectation of usage. Lastly, the host bus interface usage is fairly unpredictable so the bus interface unit 110 is allocated the remaining access time.

In another embodiment of the present invention, two or more units are allocated access time outside the scope of the global arbitration limit. In still another embodiment, no units are allocated access time outside the scope of the global arbitration limit. Rather, all of the units are granted access within the global arbitration limit and thus the global arbitration limit alone defines the maximum access cycle time. In still other embodiments, the number of units limited under the global arbitration limit can be varied. Optionally, additional global arbitration limits can be

added that operate concurrently or sequentially. Other embodiments are contemplated in which additional units may access the buffer memory 118 in a manner similar to the disk formatter 116, outside of the global arbitration limit.

While certain exemplary preferred embodiments have been described and shown in the accompanying drawings, it is to be understood that such embodiments are merely illustrative of and not restrictive on the broad invention. Further, it is to be understood that this invention shall not be limited to the specific construction and arrangements shown and described since various modifications or changes may occur to those of ordinary skill in the art without departing from the spirit and scope of the invention as claimed. It is intended that the scope of the invention be limited not by this detailed description but by the claims appended hereto.

**WHAT IS CLAIMED IS:**

1. A method for providing access to a buffer memory of a hard disk controller comprising the acts of:

allocating a first amount of time to an error correction unit;

allocating a second amount of time to a host interface unit;

accessing said buffer memory using said error correction unit for a first duration, said first duration being no greater than said first amount of time;

adjusting said second amount of time by an amount related to the difference between said first amount of time and said first duration; and

accessing said buffer memory using said host interface unit for a second duration, said second duration being no greater than said second amount of time.

2. The method as defined in Claim 1, further comprising the acts of:

allocating a third amount of time to a disk formatter unit;

granting buffer memory access to said disk formatter unit; and

accessing said buffer memory using said disk formatter unit for a third duration, said third duration being no greater than said third amount of time, wherein any unused portion of said third amount of time is not allocated to said host interface unit.

3. The method as defined in Claim 2, wherein the disk formatter unit is again granted buffer memory access within a predetermined fourth amount of time after the end of said third duration.

4. The method as defined in Claim 1, wherein access to said buffer memory is provided during a repeating access cycle during which no unit accesses said buffer memory more than once.

5. A method for providing access to a shared controller resource of a mass storage device controller comprising the acts of:

providing a first access cycle during which access to said shared controller resource is arbitrated;

allocating a first portion of said first access cycle as a first continuous block of access time to a first accessing unit;

utilizing an amount of said first continuous block of access time using said first accessing unit;

allocating a second portion of said first access cycle and an unused amount of said first continuous block of access time as a second continuous block of access time to a second accessing unit; and

utilizing an amount of said second continuous block of access time using said second accessing unit.

6. The method as defined in Claim 5, further comprising the acts of:

allocating a third portion of said first access cycle as a third continuous block of access time to a third accessing unit;

granting resource access to said third accessing unit during said first access cycle;

utilizing an amount of said third continuous block of access time with said third accessing unit during said first access cycle;

releasing access of said resource by said third accessing unit; and

granting resource access to said third accessing unit during a second access cycle, wherein said second access cycle grant is performed within a predetermined amount of time after said releasing act.

7. The method as defined in Claim 6, further comprising the acts of:

specifying a duration associated with said access cycle; and

reducing the duration of at least one iteration of said access cycle by an amount related to an unused amount of said second continuous block of access time.

8. The method as defined in Claim 6, wherein said shared controller resource is a buffer memory.

9. The method as defined in Claim 6, wherein said first accessing unit is a processor and said second accessing unit is a host interface circuit.

10. A method of varying a duration of an access cycle of a mass storage device controller shared resource, wherein the access of multiple units to the shared controller resource is arbitrated, said method comprising the acts of:

providing an access cycle, said access cycle having a cycle duration;

allocating a first amount of time of said cycle duration to a first accessing unit;

accessing said shared controller resource using said first unit for a first duration, said first duration being no greater than said first amount of time; and

allocating a second amount of time and an unused amount of said first amount of time to a second accessing unit, wherein said unused amount of said first amount of time is based on a difference between said first amount of time and said first duration.

11. The method as defined in Claim 10, further comprising the acts of:

allocating a third amount of time to a third accessing unit;

accessing said shared controller resource using said third unit for a third duration, said third duration being no greater than said third amount of time; and

reducing said duration of said access cycle by an amount related to an unused amount of said third amount of time.

12. The method as defined in Claim 10, wherein said shared controller resource is a controller memory.

13. The method as defined in Claim 11, wherein said first accessing unit is a memory refresh circuit, said second accessing unit is a host interface circuit, and said third accessing unit is a disk formatter circuit.

14. A method for providing access to a shared memory of a disk controller, said method comprising the acts of:

allocating an amount of time as an available memory access time to each of at least two units; and

accessing said shared controller memory using said at least two units, wherein unused time allocated to a first of said at least two units is reallocated to a second of said at least two units.

15. The method as defined in Claim 14, wherein said at least two units includes at least a data fetch circuit and a host interface circuit.

16. A method for providing access to a shared controller resource of a mass storage device controller comprising the acts of:

allocating a first amount of time to a first accessing unit;

allocating a second amount of time to a second accessing unit;

accessing said shared controller resource using said first unit for a first duration, said first duration being no greater than said first amount of time; and

adjusting said second amount of time by an amount related to the difference between said first amount of time and said first duration; and

accessing said shared controller resource using said second unit for a second duration, said second duration being no greater than said adjusted second amount of time.

17. A mass storage device controller circuit, including a memory shared by multiple circuits, said controller circuit comprising:

an error correction circuit used to correct data read from a mass storage device, said error correct circuit associated with a first amount of time allocated to access said shared memory;

a host system interface circuit used to exchange data with a host system, said host system interface circuit associated with a second amount of time allocated to access said shared memory;

a storage device formatter circuit used to interface to said mass storage device, said storage device formatter circuit associated with a third amount of time allocated to access said shared memory; and

an arbiter circuit used to arbitrate access to said shared memory by at least said host system interface circuit, said error correction circuit, and said storage device formatter circuit, wherein said arbitration circuit is configured to grant said error correction circuit shared memory access for a first

duration, said first duration being no greater than said first amount of time, and to adjust said second amount of time by an amount related to the difference between said first amount of time and said first duration granting buffer shared memory access to said disk formatter unit,

wherein said arbiter circuit grants said storage device formatter circuit shared memory access for a third duration, said third duration being no greater than said third amount of time.

18. The mass storage device controller circuit as defined in Claim 17, wherein said arbiter is further configured to provide access to said shared memory during a repeating access cycle during which no circuit accesses the shared memory more than once.

19. The mass storage device controller circuit as defined in Claim 17, wherein said storage device formatter circuit is provided access to said shared memory within a predetermined time period after a previous access of said shared memory by said storage device formatter circuit.

20. The mass storage device controller circuit as defined in Claim 17, wherein said arbiter is a state machine.

21. A computer system comprising:

a host system;

a host system interface circuit coupled to said host system, said host system interface circuit used to exchange data with said host system;

an error correction circuit used to correct erroneous data read from a mass storage device;

a storage device formatter circuit used to interface to said mass storage device; and

an arbiter circuit used to arbitrate access to a buffer memory by at least said host system interface circuit, said error correction circuit, and said storage device formatter circuit, wherein said arbiter circuit is configured to allocate a first amount of time to said error correction circuit, to allocate a second amount of time to said host interface circuit, and to allocate a third amount of time to said storage device formatter circuit, and said arbiter circuit is further configured to provide said error correction circuit access for a first duration, said first duration being no greater than said first amount of time, to adjust said second amount of time by an amount related to the difference between said first amount of time and said first duration, and to provide said storage device formatter circuit access for a third duration no greater than said third amount of time.

22. The computer system as defined in Claim 21, wherein said arbiter circuit is further configured to provide access to said shared memory during a repeating access cycle during which no circuit accesses the shared controller resource more than once.

23. A mass storage device controller comprising:

a means for buffering data;

a means for exchanging data with a host system, said means for exchanging data with a host system coupled to said means for buffering data;

a means for processing data coupled to said means for buffering data;

a means for exchanging data with a mass storage device, said means for exchanging data with said mass storage device coupled to said means for buffering data; and

a means for arbitrating access to said means for buffering data by at least said means for exchanging data with a host system, said means for processing data, and said means for exchanging data with a mass storage device, wherein said means for arbitrating access is configured to allocate a first amount of time to said means for processing data, to allocate a second amount of time to said means for exchanging data with a host system, and to allocate a third amount of time to said means for exchanging data with a mass storage device, wherein said means for arbitrating is further configured to provide said means for processing data access for a first duration, said first duration being no greater than said first amount of time, to adjust said second amount of time by an amount related to the difference between said first amount of time and said first duration, and to provide said means for exchanging data with a mass storage device access for a third duration no greater than said third amount of time.

24. The mass storage device controller as defined in Claim 23, wherein said means for arbitrating access is further configured to provide access to said means for buffering data during a repeating access cycle during which no accessing means accesses said means for buffering data more than once.

25. A method for providing access to a shared controller resource of a mass storage device controller comprising the acts of:

allocating a first amount of time as a first available access time to a first accessing unit;

allocating a second amount of time as a second available access time to a second accessing unit, said second available access time including at least a portion of said first available access time;

accessing said shared controller resource using said first unit for a first duration, said first duration being no greater than said first available access time;

decreasing said second available access time by said first duration; and

accessing said shared controller resource using said second unit for a second duration, said second duration being no greater than said decreased second available access time.

26. The method as defined in Claim 25, further comprising the acts of:

allocating a third amount of time as a third available access time to a third accessing unit, said third amount of time not included in said second amount of time; and

accessing said shared controller resource using said third unit for a third duration, said third duration being no greater than said third available access time.

27. The method as defined in Claim 25, wherein access to the shared controller resource is provided during a repeating access cycle during which no unit accesses the shared controller resource more than once.

28. The method as defined in Claim 26, wherein said access cycle has a maximum duration approximately equal to the sum of said second amount of time and said third amount of time.

29. The method as defined in Claim 26, wherein said third unit has a data buffer through which data passes as it moves between said shared controller resource and said third accessing unit.

30. The method as defined in Claim 26, wherein said third amount of time is determined at least in part by the size of said data buffer, and by the difference between the rate at which data is transferred between said shared controller resource and said data buffer and the rate at which data is processed by said third accessing unit.

31. The method as defined in Claim 26, wherein said second amount of time is determined at least in part by the size of said data buffer and by the rate at which data is processed by said third accessing unit.

32. The method as defined in Claim 26, wherein said second amount of time and said third amount of time are determined at least in part by taking into account the amount of time needed to begin a transfer of data between said shared controller resource and said data buffer upon a release of access by an accessing unit.

33. The method as defined in Claim 26, wherein said second unit is a host interface unit and said third unit is a storage device formatter unit.

34. A disk controller circuit, including a memory accessed by a plurality of disk controller circuits, said disk controller circuit comprising:

a first controller circuit with a corresponding first allocated access duration;

a second controller circuit with a corresponding second allocated access duration; and

an arbiter circuit which measures a quantity associated with an actual access duration by said first controller circuit and adjusts said second allocated access duration by an amount related to a difference between said first allocated access duration and said actual access duration.

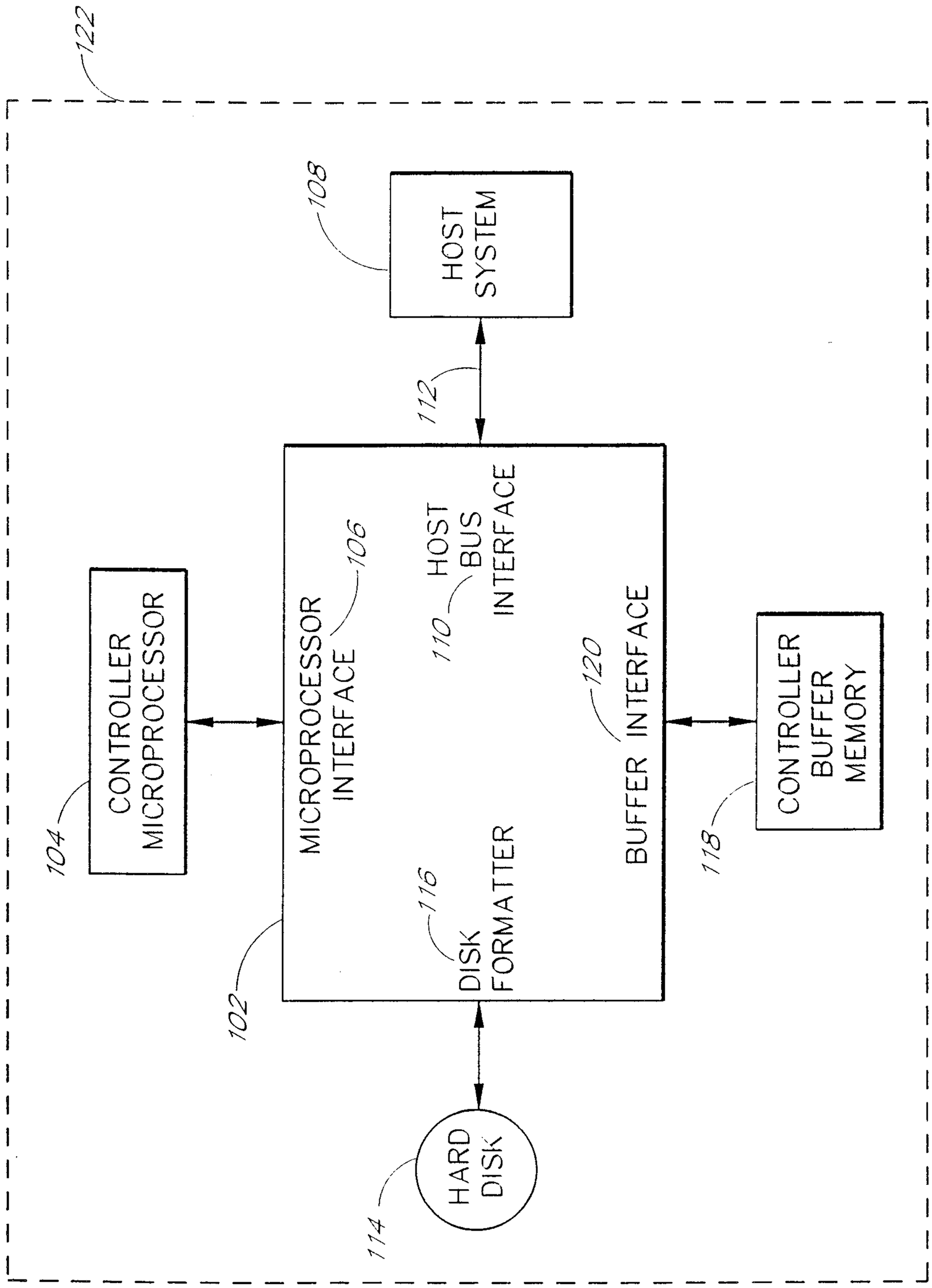

100
122
108
HOST SYSTEM
112
HOST BUS INTERFACE
110
MICROPROCESSOR INTERFACE
106
DISK FORMATTER
116
BUFFER INTERFACE
120
CONTROLLER MICROPROCESSOR
104
CONTROLLER BUFFER MEMORY
118
102
HARD DISK
114

FIG. 1

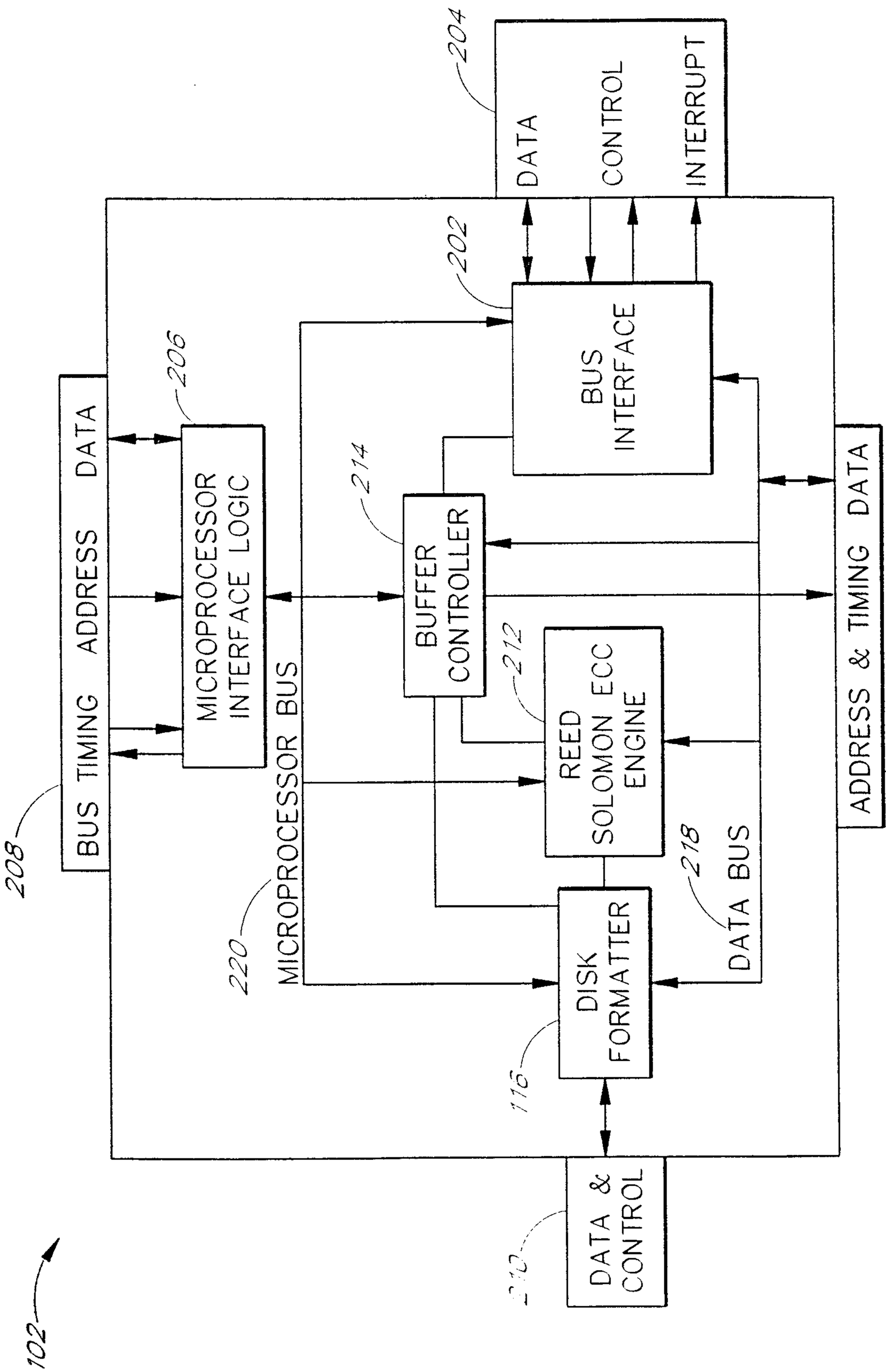

204
DATA
CONTROL
INTERRUPT
202
BUS INTERFACE
206
MICROPROCESSOR INTERFACE LOGIC
214
BUFFER CONTROLLER
212
REED SOLOMON ECC ENGINE
208
BUS TIMING
ADDRESS
DATA
220
MICROPROCESSOR BUS
218
DATA BUS
116
DISK FORMATTER
216
ADDRESS & TIMING
DATA
210
DATA & CONTROL
102

FIG. 2

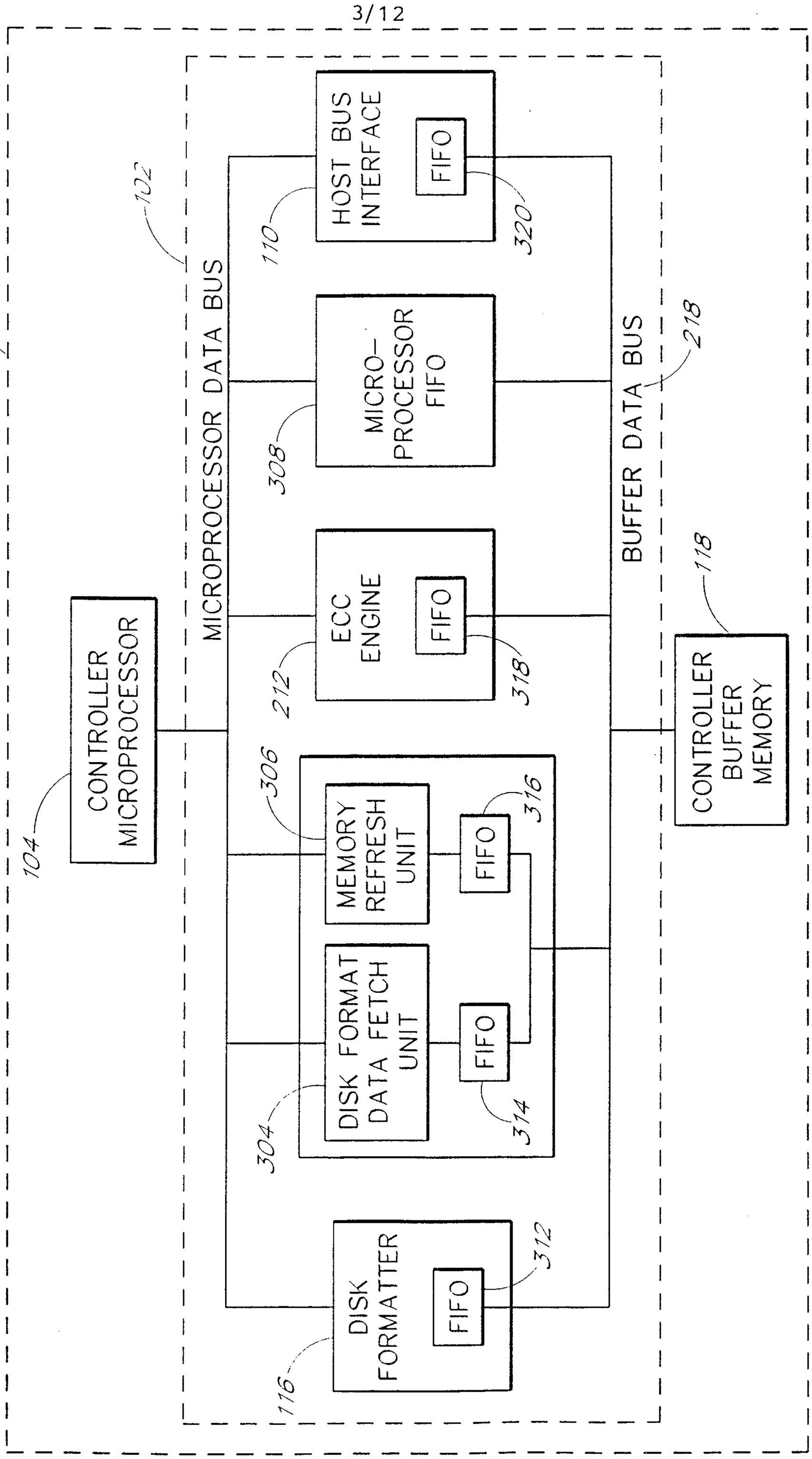

HOST BUS INTERFACE
FIFO
110
320
122
102
MICROPROCESSOR DATA BUS
MICRO-PROCESSOR FIFO
308
ECC ENGINE
FIFO
312
318
BUFFER DATA BUS
218
CONTROLLER MICROPROCESSOR
104
118
CONTROLLER BUFFER MEMORY
306
MEMORY REFRESH UNIT
FIFO
316
DISK FORMAT DATA FETCH UNIT
FIFO
304
314
DISK FORMATTER
FIFO
312
116

FIG. 3

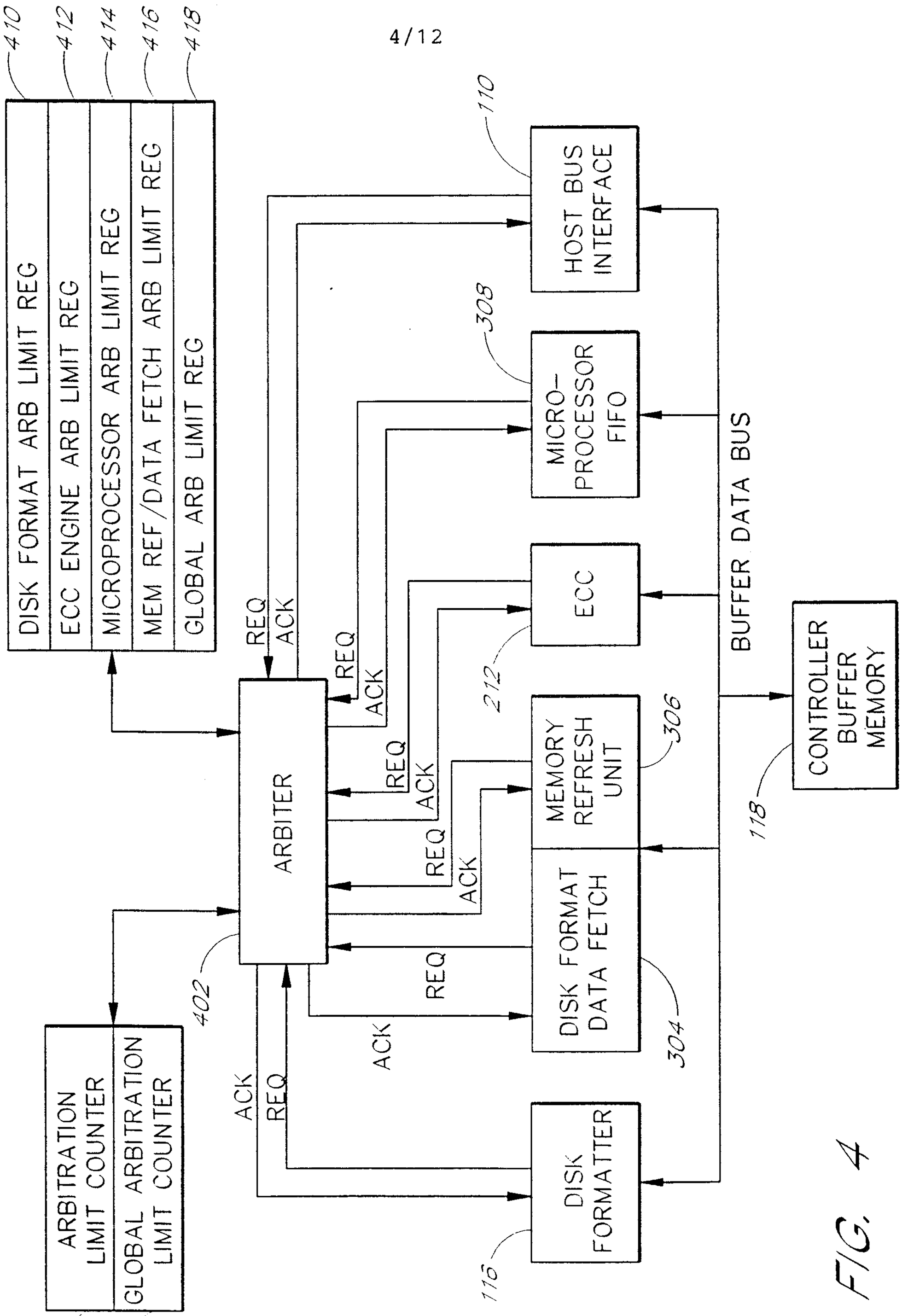

410
412
414
416
418
DISK FORMAT ARB LIMIT REG
ECC ENGINE ARB LIMIT REG
MICROPROCESSOR ARB LIMIT REG
MEM REF/DATA FETCH ARB LIMIT REG
GLOBAL ARB LIMIT REG
110
HOST BUS INTERFACE
308
MICRO-PROCESSOR FIFO
212
ECC
306
MEMORY REFRESH UNIT
304
DISK FORMAT DATA FETCH
BUFFER DATA BUS
118
CONTROLLER BUFFER MEMORY
402
ARBITER
REQ
ACK
116
DISK FORMATTER
406
ARBITRATION LIMIT COUNTER
408
GLOBAL ARBITRATION LIMIT COUNTER

FIG. 4

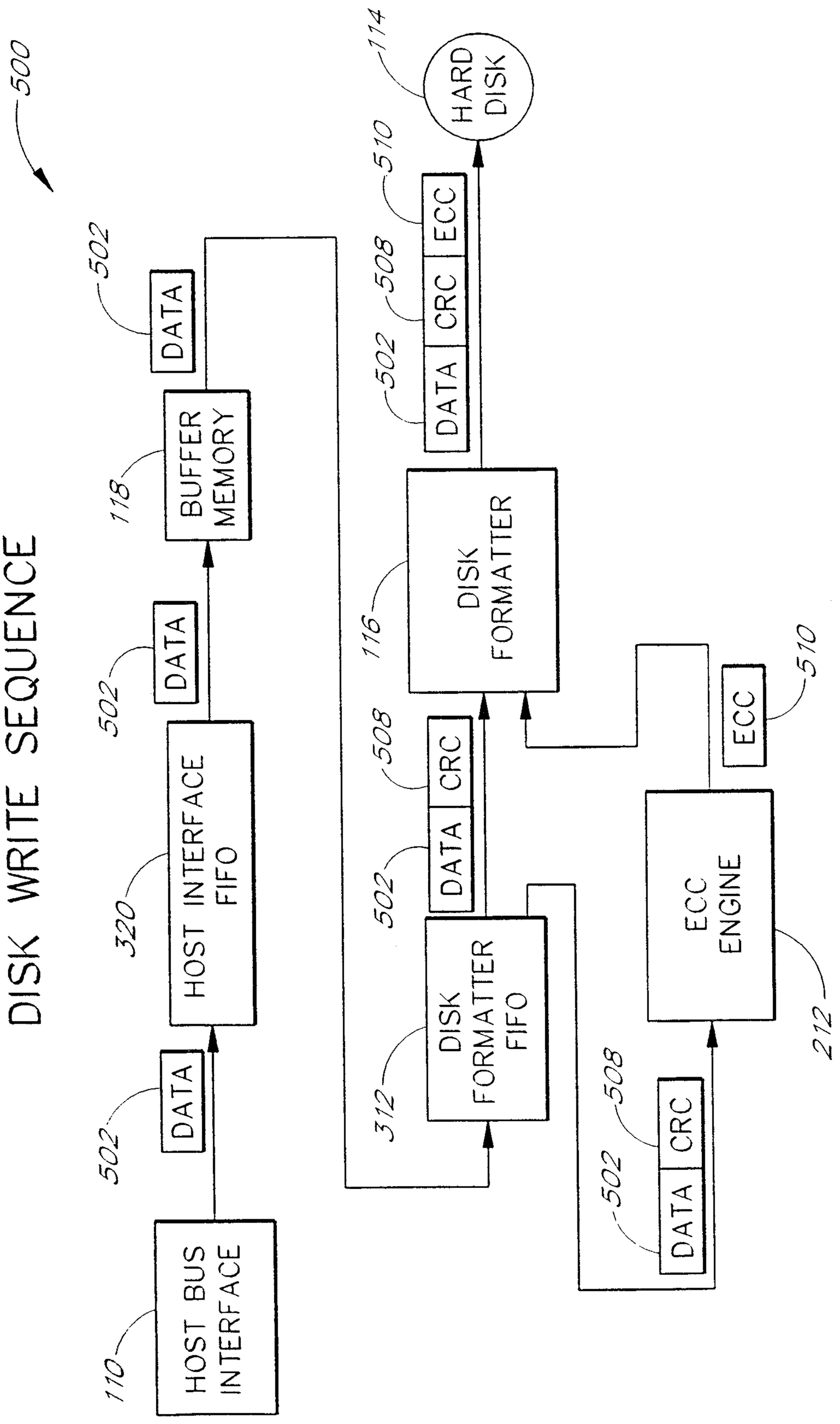

DISK WRITE SEQUENCE
500
HOST BUS INTERFACE
110
DATA
502
HOST INTERFACE FIFO
320
DATA
502
BUFFER MEMORY
118
DATA
502
CRC
508
ECC
510
HARD DISK
DISK FORMATTER
116
DATA
502
CRC
508
DISK FORMATTER FIFO
312
ECC ENGINE
212
ECC
510
DATA
502
CRC
508

FIG. 5

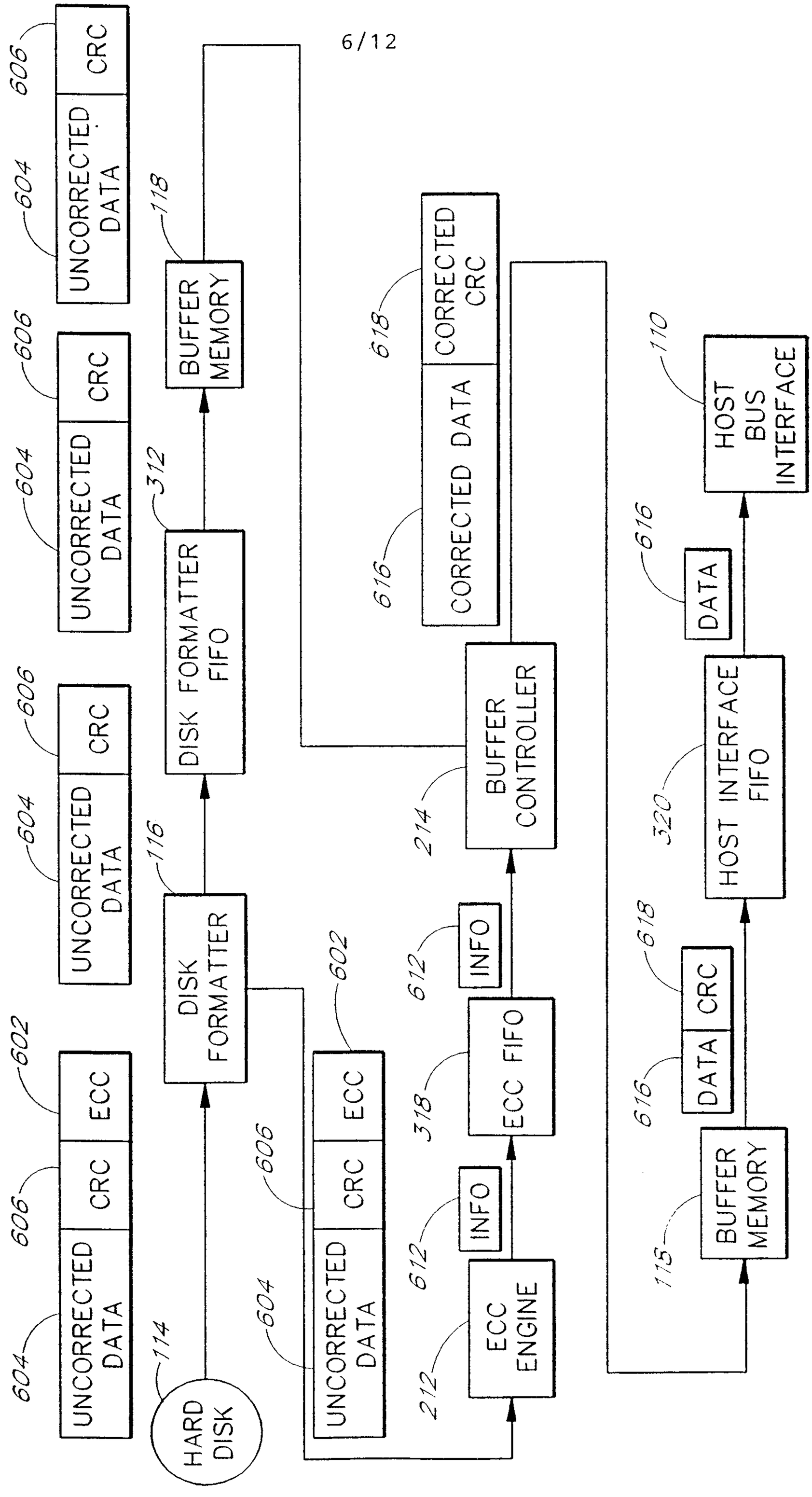

DISK READ SEQUENCE AND DATA FORMAT
602
ECC
606
CRC
604
UNCORRECTED DATA
UNCORRECTED DATA
606
CRC
604
UNCORRECTED DATA
606
CRC
604
UNCORRECTED DATA
602
ECC
606
CRC
604
UNCORRECTED DATA
114
HARD DISK
116
DISK FORMATTER
312
DISK FORMATTER FIFO
BUFFER MEMORY
118
602
ECC
606
CRC
604
UNCORRECTED DATA
612
INFO
318
ECC FIFO
212
ECC ENGINE
214
BUFFER CONTROLLER
618
CORRECTED CRC
616
CORRECTED DATA
110
HOST BUS INTERFACE
616
DATA
618
CRC
320
HOST INTERFACE FIFO
115
BUFFER MEMORY

FIG. 6

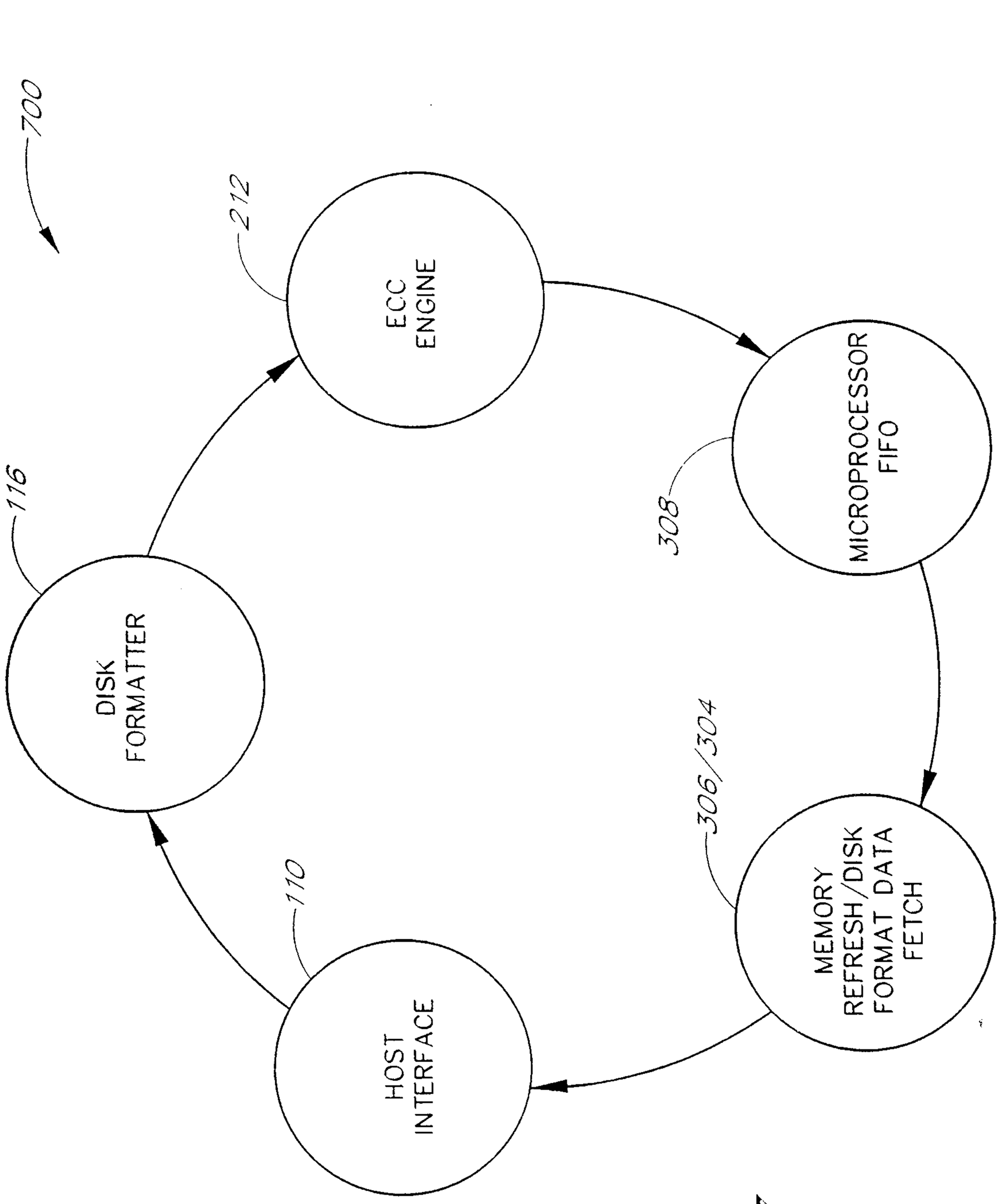

700
212
ECC ENGINE
116
DISK FORMATTER
110
HOST INTERFACE
306/304
MEMORY REFRESH/DISK FORMAT DATA FETCH
308
MICROPROCESSOR FIFO

FIG. 7

FIG. 8

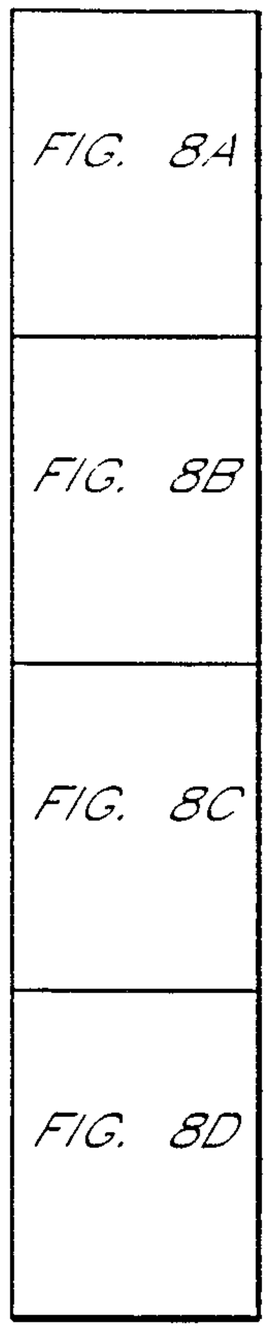
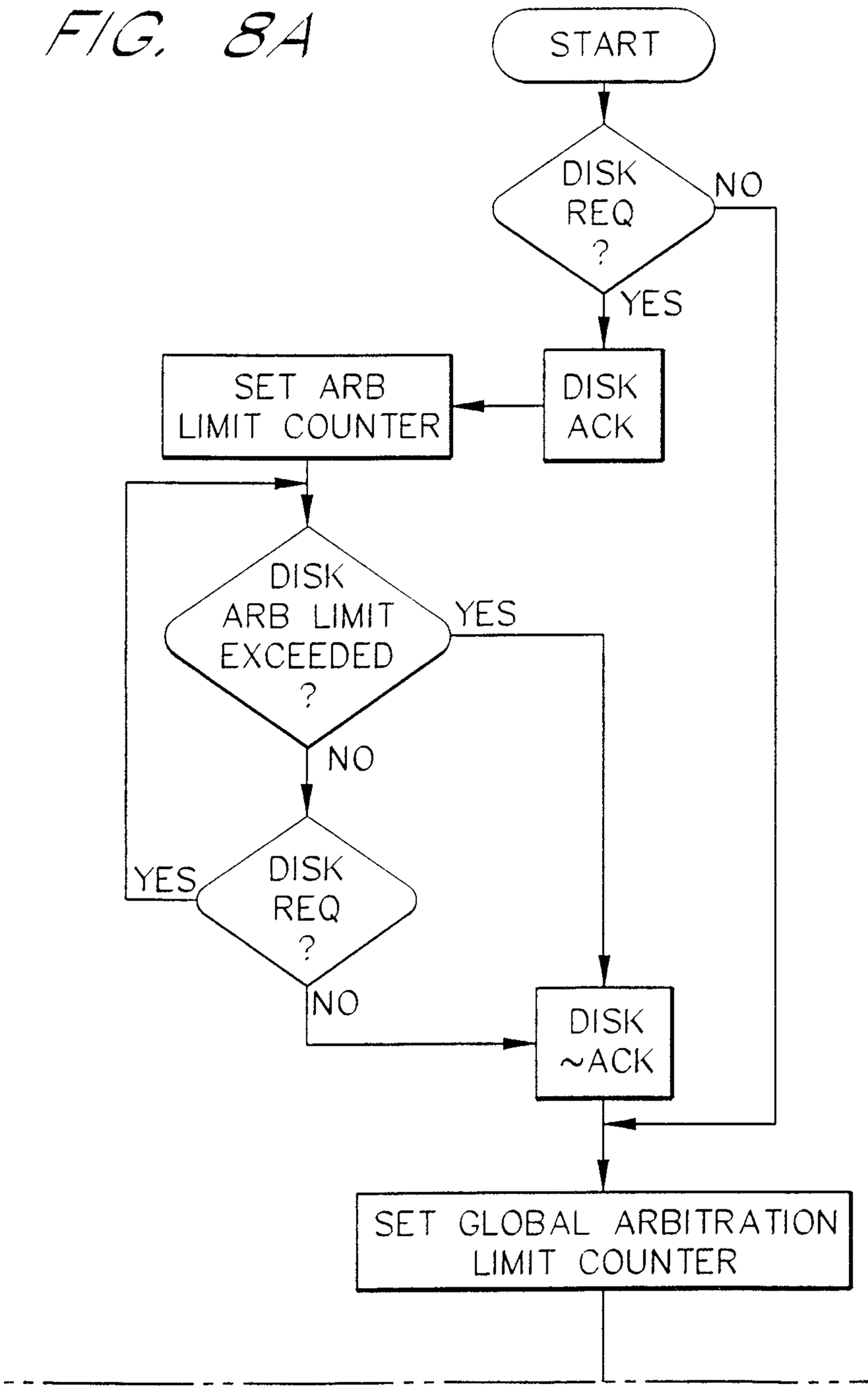


FIG. 8A
FIG. 8B
FIG. 8C
FIG. 8D

FIG. 8A


START
DISK REQ ?
NO
YES
DISK ACK
SET ARB LIMIT COUNTER
DISK ARB LIMIT EXCEEDED ?
YES
NO
YES
DISK REQ ?
NO
DISK ~ACK
SET GLOBAL ARBITRATION LIMIT COUNTER

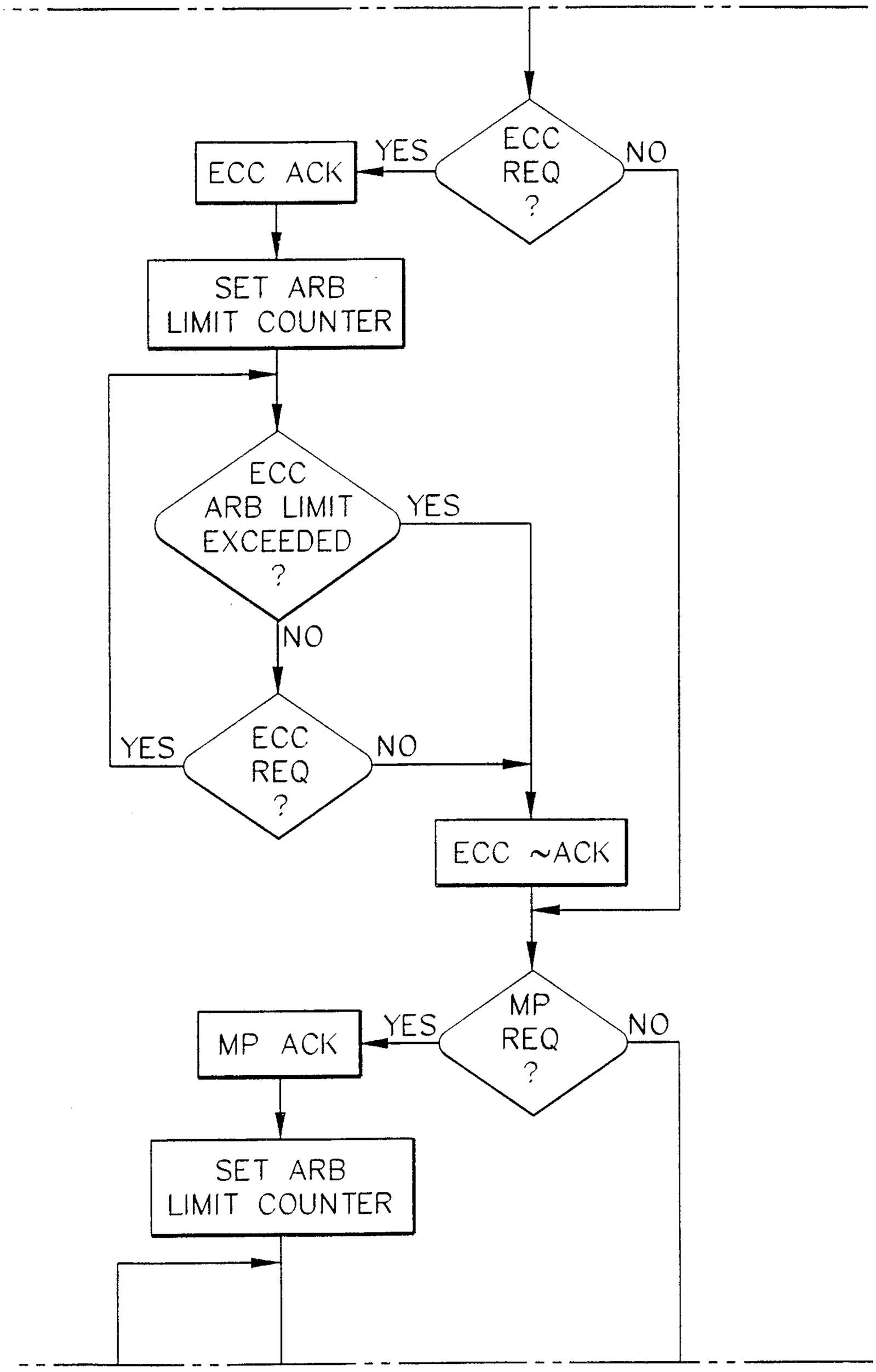

ECC REQ ?
YES
NO
ECC ACK
SET ARB LIMIT COUNTER
ECC ARB LIMIT EXCEEDED ?
YES
NO
ECC REQ ?
YES
NO
ECC ~ACK
MP REQ ?
YES
NO
MP ACK
SET ARB LIMIT COUNTER

FIG. 8B

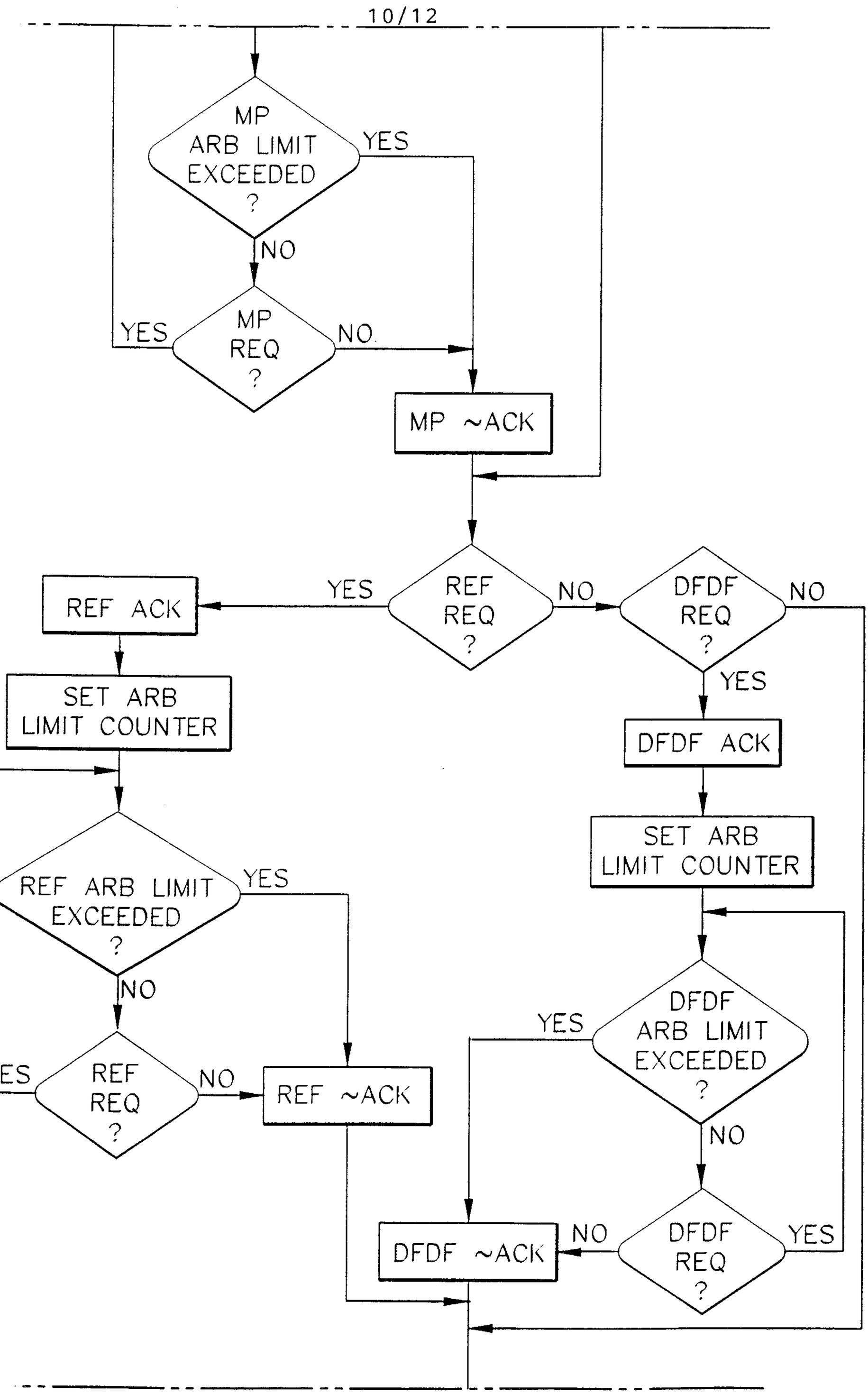

MP ARB LIMIT EXCEEDED ?
YES
NO
MP REQ ?
YES
NO
MP ~ACK
REF REQ ?
YES
NO
REF ACK
DFDF REQ ?
NO
YES
SET ARB LIMIT COUNTER
DFDF ACK
REF ARB LIMIT EXCEEDED ?
YES
NO
SET ARB LIMIT COUNTER
REF REQ ?
YES
NO
REF ~ACK
DFDF ARB LIMIT EXCEEDED ?
YES
NO
DFDF ~ACK
DFDF REQ ?
NO
YES

FIG. 8C

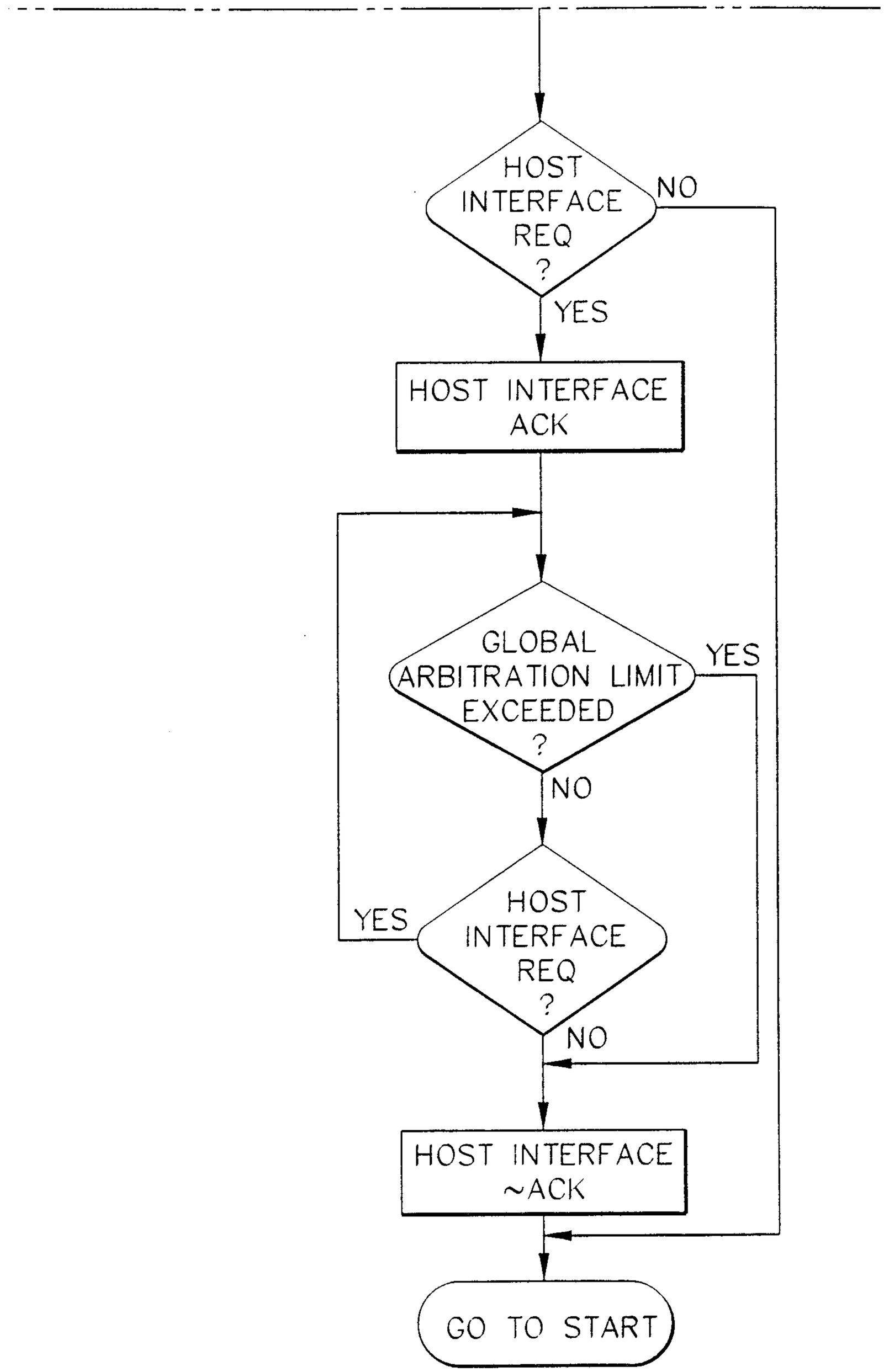

HOST INTERFACE REQ ?
NO
YES
HOST INTERFACE ACK
GLOBAL ARBITRATION LIMIT EXCEEDED ?
YES
NO
HOST INTERFACE REQ ?
YES
NO
HOST INTERFACE ~ACK
GO TO START

FIG. 8D

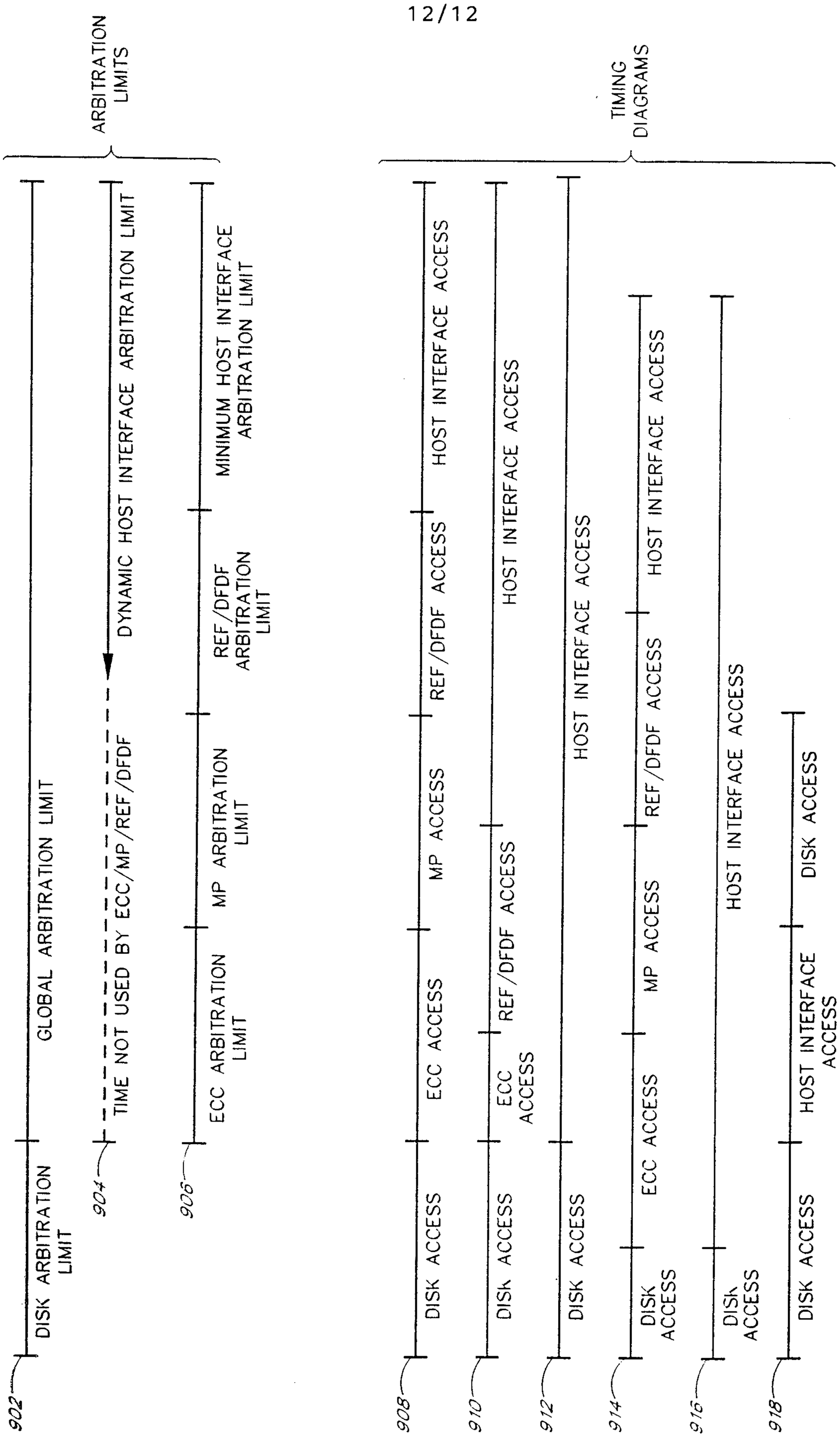
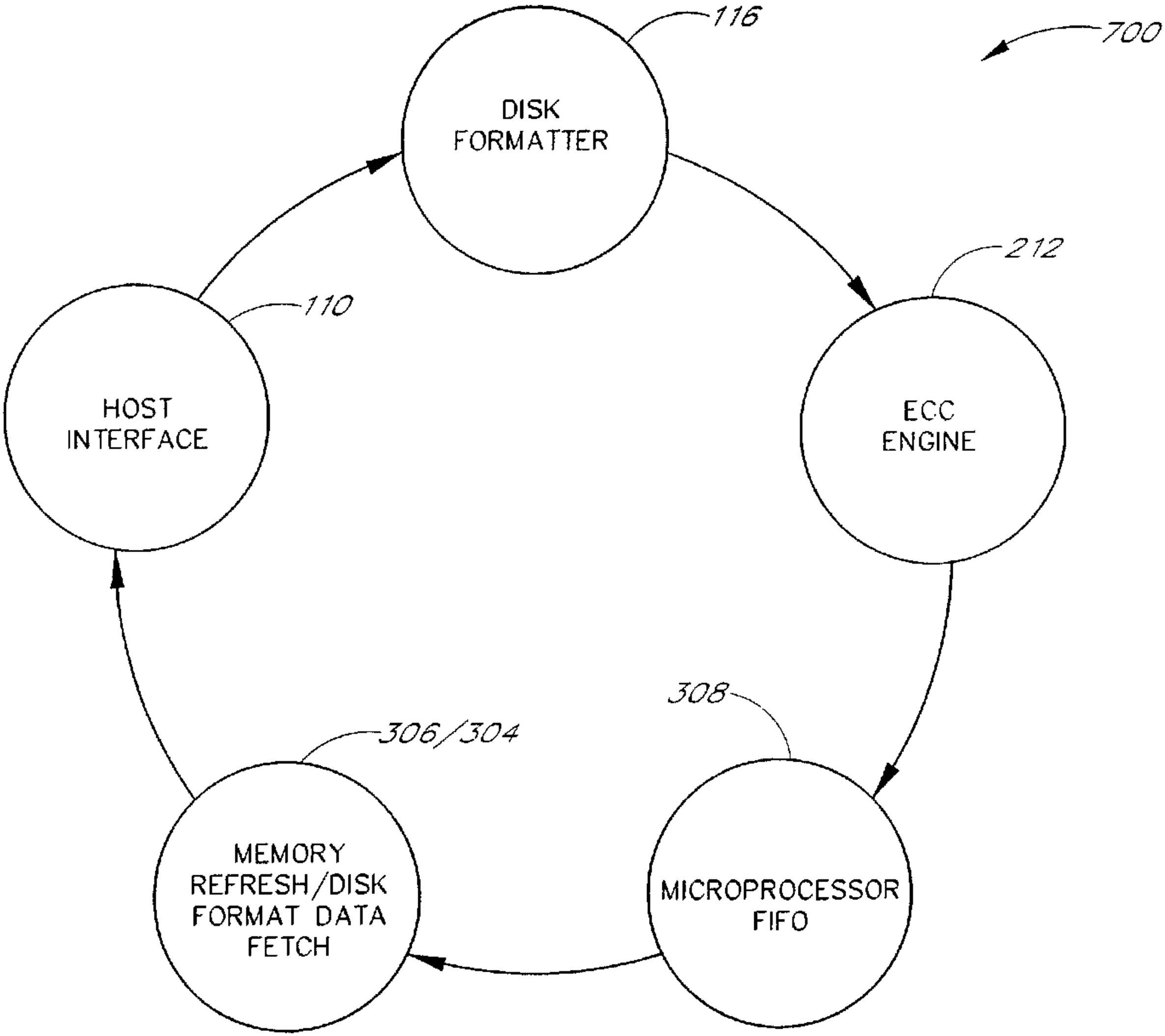

ARBITRATION LIMITS
TIMING DIAGRAMS
GLOBAL ARBITRATION LIMIT
DYNAMIC HOST INTERFACE ARBITRATION LIMIT
TIME NOT USED BY ECC/MP/REF/DFDF
MINIMUM HOST INTERFACE ARBITRATION LIMIT
REF/DFDF ARBITRATION LIMIT
ECC ARBITRATION LIMIT
MP ARBITRATION LIMIT
DISK ARBITRATION LIMIT
902
904
906
HOST INTERFACE ACCESS
REF/DFDF ACCESS
MP ACCESS
ECC ACCESS
DISK ACCESS
908
910
912
914
916
918

FIG. 9


700
116
DISK FORMATTER
110
HOST INTERFACE
212
ECC ENGINE
306/304
MEMORY REFRESH/DISK FORMAT DATA FETCH
308
MICROPROCESSOR FIFO