**PCT** WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

**(51) International Patent Classification 5 :** G06F 12/08, 3/06, 11/10

**A1**

**(11) International Publication Number:** WO 92/04674

**(43) International Publication Date:** 19 March 1992 (19.03.92)

**(21) International Application Number:** PCT/GB91/01557

**(22) International Filing Date:** 12 September 1991 (12.09.91)

**(30) Priority data:**
9019891.2 12 September 1990 (12.09.90) GB

**(71) Applicant** *(for all designated States except US):* HI-DATA LIMITED [GB/GB]; Unit 8, The Old Mill, 61 Reading Road, Pangbourne, Berkshire RG8 7HY (GB).

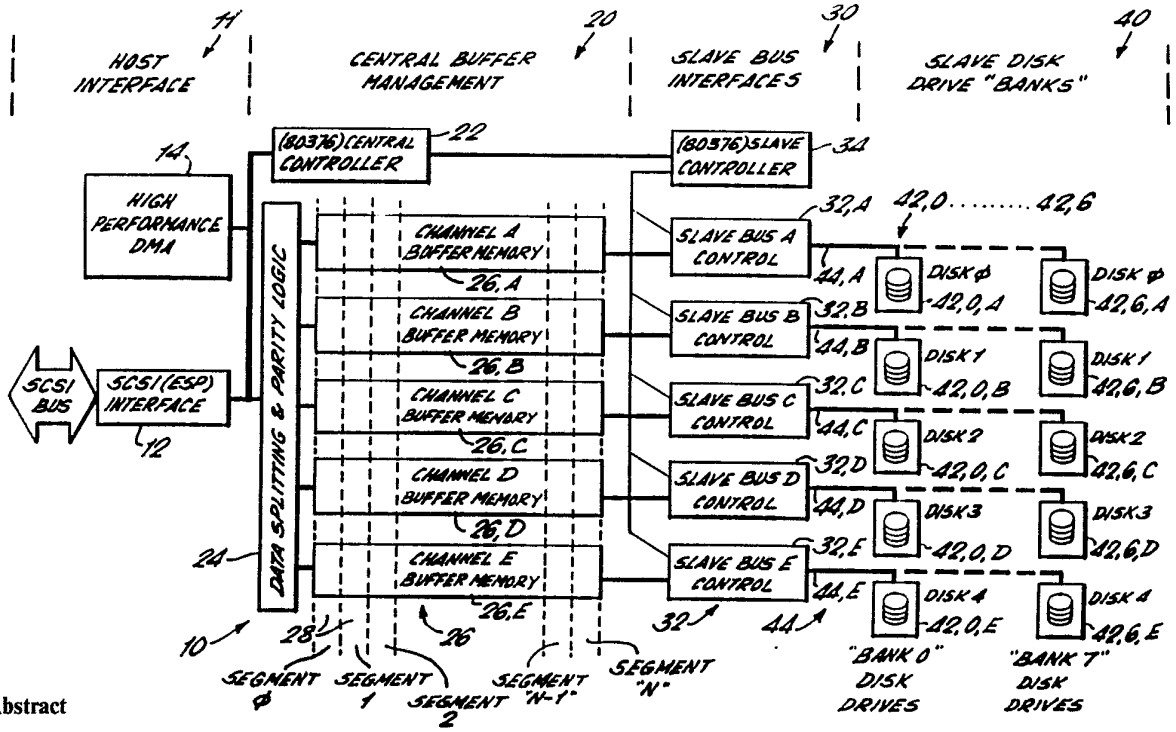
**(72) Inventor; and**
**(75) Inventor/Applicant** *(for US only) :* HILL, Andrew, James, William [GB/GB]; 3 Thomson Walk, Calcot Heights, Reading RG3 7DP (GB).

**(74) Agent:** BOULT WADE & TENNANT; 27 Furnival Street, London EC4A 1PQ (GB).

**(81) Designated States:** AT (European patent), AU, BE (European patent), CA, CH (European patent), DE (European patent), DK (European patent), ES (European patent), FR (European patent), GB, GB (European patent), GR (European patent), IT (European patent), JP, LU (European patent), NL (European patent), SE (European patent), US.

**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:** COMPUTER MEMORY ARRAY CONTROL




**(57) Abstract**

A controller for and method of controlling the transfer of data between a host computer and a number of memory units (42), particularly magnetic disk drives is disclosed. A request for data by the host computer causes the central controller (22) to check the buffer (26) formed of buffer segments (28) to establish whether the requested data is contained therein. If so the data is supplied to the host computer. If the requested data is not present in the buffer (26) data is read from the memory units (42) and supplied to the host computer. In addition further data which is logically sequential to the requested data is stored in a buffer segment (28). The central controller (22) operates to control the size and number of the buffer segments (28).

*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.

| | | | | | |
|---|---|---|---|---|---|
| **AT** | Austria | **ES** | Spain | **MG** | Madagascar |
| **AU** | Australia | **FI** | Finland | **ML** | Mali |
| **BB** | Barbados | **FR** | France | **MN** | Mongolia |
| **BE** | Belgium | **GA** | Gabon | **MR** | Mauritania |
| **BF** | Burkina Faso | **GB** | United Kingdom | **MW** | Malawi |
| **BG** | Bulgaria | **GN** | Guinea | **NL** | Netherlands |
| **BJ** | Benin | **GR** | Greece | **NO** | Norway |
| **BR** | Brazil | **HU** | Hungary | **PL** | Poland |
| **CA** | Canada | **IT** | Italy | **RO** | Romania |
| **CF** | Central African Republic | **JP** | Japan | **SD** | Sudan |
| **CG** | Congo | **KP** | Democratic People's Republic of Korea | **SE** | Sweden |
| **CH** | Switzerland | | | **SN** | Senegal |
| **CI** | Côte d'Ivoire | **KR** | Republic of Korea | **SU**+ | Soviet Union |
| **CM** | Cameroon | **LI** | Liechtenstein | **TD** | Chad |
| **CS** | Czechoslovakia | **LK** | Sri Lanka | **TG** | Togo |
| **DE*** | Germany | **LU** | Luxembourg | **US** | United States of America |
| **DK** | Denmark | **MC** | Monaco | | |

**+ Any designation of "SU" has effect in the Russian Federation. It is not yet known whether any such designation has effect in other States of the former Soviet Union.**

# COMPUTER MEMORY ARRAY CONTROL

This invention relates to computer memories, and in particular to a controller for controlling and a method of controlling an array of memory units in a computer.

For high performance Operating Systems and Fileservers, an idealistic computer memory would be a memory having no requirement to "seek" the data. Such a memory would have instantaneous access to all data areas. Such a memory could be provided by a RAM disk. This would provide for access to data regardless of whether it was sequential or random in its distribution in the memory. However, the use of RAM is disadvantageous compared to the use of conventional magnetic disk drive storage media in view of the high cost of RAM and especially due to the additional high cost of providing "redundancy" to compensate for failure of memory units.

Thus the most commonly used non-volatile computer memories are magnetic disk drives. However, these disk drives suffer from the disadvantage that they require a period of time to position the head or heads with the correct part of the disk corresponding to the location of the data. This is termed the seek and rotation delay. This delay becomes a significant portion of the data access time when only a small amount of data is to be read or written to or from the disk.

For disk drives having a large capacity, the seek time can considerably limit the operating speed of a computer. The input/output (I/O) speed of disk drives has not kept pace with the development of microprocessors and therefore memory access time can severely restrain the performance of modern computers.

In today's practical environment, with modern operating systems, data files tend to be sequential in the nature of their storage on the disk drive surface. Also, read and write operations tend to be sequential, or at least partially sequential in their nature. Therefore if the seek operations could be reduced or eliminated within the disk drives whenever access to the sequential data area is required, considerable performance enhancements would be achieved. With the seek operations eliminated, the data would appear to come from a very fast access system whose data rate was controlled by the data rate of the disk drive being utilised.

In order to reduce the data access time for a large memory, a number of industry standard relativity inexpensive disk drives have been used. Since a large array of these is used, some redundancy must be incorporated in the array to compensate for disk drive failure.

It is known to provide disk drives in an array of drives in such a way that the contents of any one drive can, should that drive fail, be reconstructed in a replacement drive from the information stored in the other drives.

Various classifications of arrangements that can perform this are described in more detail in a paper by D.A. Patterson, G. Gibson and R.H. Katz under the title "A Case for Redundant Arrays of Inexpensive Disks (RAID)", Report No. UCB/CSD 87/391 12/1987, Computer Science Division, University of California, U.S.A., the content of which is incorporated herein by reference.

This document describes two types of arrangements. The first of these arrangements is particularly adapted for large scale data transfers and is termed "RAID-3". In this

arrangement at least three disk drives are provided in which sequential bytes of information are stored in the same logical block positions on the drives, one drive having a check byte created by a controller written thereto, which enables any one of the other bytes on the disk drives to be determined from the check byte and the other bytes. The term "RAID-3" as used hereinafter is as defined by the foregoing passage.

In the RAID-3 arrangement there is preferably at least five disk drives, with four bytes being written to the first four drives and the check byte being written to the fifth drive, in the same logical block position as the data bytes on the other drives. Thus, if any drive fails, each byte stored on it can be reconstructed by reading the other drives. Not only can the computer be arranged to continue to operate despite failure of a disk drive, but also the failed disk drive can be replaced and rebuilt without the need to restore its contents from probably out-of-date backup copies. Moreover, even if one drive should fail, there is no loss of performance of the computer while the failed disk drive remains inactive and while it is replaced. A disk drive storage system having the RAID-3 arrangement is described in EP-A-0320107, the content of which are incorporated herein by reference.

The second type of storage system which is particularly adapted for multi-user applications, is termed "RAID-5". In the RAID-5 arrangement there are preferably at least five disk drives in which four sectors of each disk drive are arranged to store data and one sector stores check information. The check information is derived not from the data in the four sectors on the disk, but from designated sectors on each of the other four disks. Consequently each disk can be rebuilt from the data and check information on the remaining disks.

RAID-5 is seen to be advantageous, at least in theory, because it allows multi-user access, albeit with equivalent transfer performance of a single disk drive.

However, a write of one sector of information involves writing to two disks, that is to say writing the information to one sector on one disk drive and writing check information to a check sector on a second disk drive. However, writing the check sector is a read modify write operation, that is, a read of the existing data and check sectors first, because the old contents of those sectors must be known before the correct check information, based on the new data to be written, can be generated and written to disk. Nevertheless, RAID-5 does allow simultaneous reads by multiple users from all disks in the system which RAID-3 cannot support.

On the other hand, RAID-5 cannot match the rate of data transfer achievable with RAID-3, because with RAID-3, both read and write operations involve a transfer to each of the five disks (in five disk systems) of only a quarter of the total amount of information transferred. Since each referral can be accomplished simultaneously the process is must faster than reading or writing to a single disk particularly where large scale transfers are involved. This is because most of the time taken to effect a read or write in respect of a given disk drive, is the time taken for the read/write heads to be positioned with resect of the disk, and for the disk to rotate to the correct angular position. Clearly, this is as long for one disk, as it is for all four. But once in the correct position, transfers of large amounts of sequential information can be effected relatively quickly.

Moreover, with the current trend for sequential information to be requested by the user, RAID-5 only offers multiple user access in theory, rather than in practice,

because requests for sequential information by the same user usually involves reading several disks in turn, thereby occupying those disks so that they are not available to other users.

Furthermore, when a drive fails in RAID-5 format, the performance of the computer is severely retarded. When reading, if the required information is on a sector in the failed drive, it must be derived by reading all four of the other disks. Similarly, when writing either check or information data to a working drive, the four working disks must first be read before the appropriate information sector is written and before the appropriate check information is determined and written.

A further problem with RAID-3 is that disk drives are presently made to read or write minimum amounts of information on each given occasion. This is the formatted sector size of the disk drive and there is usually a minimum of 256 Bytes. In RAID-3 format this means that the minimum block length on any read or write is 1,024 Bytes. With growing disk drive capacities the tendency is towards even larger minimum block sizes such as 512 Bytes, so that RAID-3 effectively quadruples that minimum to 2,048 Bytes. However, many applications for computers, for example those employing UNIX version 5.3 require a minimum block size of only 512 Bytes and in this event, the known RAID-3 technique is not easily available to such systems. RAID-5 on the other hand does not increase the minimum data block size.

Nevertheless, it is the multi-user capability of RAID-5 which makes it theoretically more advantageous than RAID-3; but, in fact, it is the data transfer rate and continued performance in the event of drive failure in RAID-3 format which gives the latter much greater potential. And so it is an object of the present invention to provide a system which exhibits the same multi-user

capability of a RAID-5 disk array, or indeed better capability in that respect, which at the same time does not suffer the disadvantages of RAID-5 when it comes to large scale sequential data transfers or when a disk drive fails; in other words, a system which offers the same if not better performance as RAID-3 and RAID-5 in combination, a combination which may be termed "RAID-35".

This object is achieved by recognition of a basic principle of current popular computer storage operation: namely, that even with multi-user access to a disk storage medium, each user normally requires some sequential data in sequential requests. That is to say, a subsequent request for data by a given user is generally, albeit not always, a request for data which logically follows, in terms of its position on the disk, the information previously requested.

In one aspect the present invention provides a computer memory controller for interfacing to a host computer comprising a buffer means for interfacing to at least one memory unit and for holding data read thereto or therefrom; said buffer means controlled to form a plurality of buffer segments for addressably storing data requested by said host computer and further data which is logically sequential thereto; and control means operative to control the transfer of data to said host computer in response to requests therefrom by first addressing said buffer segments to establish whether the requested data is contained therein and if so supplying said data to said host computer, and if the requested data is not contained in the buffer segments reading said data from the or each memory unit, supplying said data to said host computer, reading from the or each memory unit further data which is logically sequential to the data requested by said host computer and storing said further data in a buffer segment; said control means controlling the buffer means to control the number and size of said buffer segments.

In another aspect the present invention provides a method of controlling an array of memory units for use with a host computer comprising the steps of receiving from said host computer a read request for data stored on the memory units, checking a plurality of buffer segments to establish whether the requested data is in said buffer segments, either complying with said request by transferring the data in said buffer segments to said host computer, or first reading said data from said memory units into one buffer segment and then complying with said request, reading from the memory units further data logically sequential to the data requested and storing said data in said buffer segment.

In another aspect, the present invention provides a computer memory controller for a host computer comprising buffer means for interfacing to at least three memory units arranged in parallel and for holding information read from said memory units; a logic circuit connected to said buffer means to recombine bytes or groups of bits successively read from successive ones of a group of said memory units; parity means operative to use a check byte or group of bits read from one of said memory units to regenerate information read from said groups of memory units if one of said group of memory units fails; said buffer means being controlled to form a number of buffer segments each storing data requested by an application run on said host computer and further data which is logically sequential thereto, and a controller for controlling the transfer of data to said host computer in response to requests from said host computer by checking said buffer segments to establish whether the requested data is in said buffer segment and supplying said data to said host computer, or reading said data from said memory units, supplying said data to said host computer, reading from said memory units further data which is logically sequential to the data requested by said host computer and storing said further data in a buffer segment.

In a further aspect the present invention provides a computer memory controller for a host computer comprising buffer means for interfacing at least three memory units arranged in parallel, a logic circuit connected to said buffer means to split data input from said host computer such that successive bytes or groups of bits from said host computer are temporarily stored in said buffer means before being successively applied to successive ones of a group of said memory units, said logic circuit being further operative to recombine bytes or groups of bits successively read from successive ones of said group of said memory units into said buffer means, said logic circuit including parity means operative to generate a check byte or group of bits from said data for temporary storage in said buffer means before being stored in at least one said memory unit, and operative to use said check byte to regenerate said data read from said group of memory units if one of said group of memory units fails, said buffer means being divided into a number of channels corresponding to the number of memory units, each said channel being divided into associated portions of buffer segments, buffer segments containing successive bytes or groups of bits corresponding to data for an application being run by said host computer, and control means operative to control the transfer of data and check bytes or groups of bits to and from said memory units in response to commands from said host computer, and operative to control the number and size of said buffer segments.

Thus, when data is requested by the host computer from the memory array, much more than what is requested is read into one of the buffer segments. Such data is termed "read-ahead data". Since computers tend to request

sequential data, particularly those running UNIX 5.4 Operating Systems and many modern Fileservers and Operating Systems, the chances are, that at a subsequent request, the requested data will actually be in the buffer, and so another read of the disk drive can be dispensed with. Indeed, it is a requirement of the computer operating system and/or the application programs being run by the various users, that in order to benefit from the present invention, the system or programs must make a habit of making at least one subsequent request for sequential data. Otherwise the present invention cannot realise the object of RAID-35 type operations.

In reality, many simultaneous application programs are run at the same time, all requiring access for sequential bursts of data. In such case each application program requires access to its own portion of the overall memory. These portions are the segments of the buffer. These segments must be large enough to optimise the electrical attributes of the interfaces being utilised by the disk drives, and to optimise the method and rate of refreshing the segments.

Preferably each buffer segment is capable of holding at least 128 Kilo-Bytes.

Such a requirement comes with the use of the currently standard disk drive interface SCSI (Small Computer System Interface). The present SCSI-1 interface starts to perform acceptably with data transfers in excess of 64 Kilo-Bytes. Therefore to allow efficient buffer refresh, each buffer memory segment would require to be twice this transfer length, or 128 Kilo-Bytes.

Preferably, write data is initially stored in some of said buffer segments, which are especially assigned for this purpose, so that actual writing to disk can be achieved in background during quiet times for the disk system.

Thus, in situations where sequential requests are made by the computer operating system and/or the application programs, the present invention gives all the theoretical advantages of RAID-5 operation and operates faster, with multiple simultaneous reads and writes, but at the same time, the simultaneous data transfer rates, and the better performance on any one disk drive failure achievable by RAID-3 format.

Preferably there are five disks employed, four being data disks and one being the check disk.

Although at present the most commonly form of redundant array of inexpensive disks used utilises magnetic disk drives, the present invention is not limited to the use of such disk drives. The present invention is equally applicable to the use of any memory device which has a long seek time for data compared to the data transfer rate once the data is located. Such media could, for instance, be an optical compact disk.

In another aspect of the present invention a computer storage system comprises an array of magnetic disk drives organised in RAID-3 format having at least three channels, said array comprising a plurality of disk drives connected to each said channel, each of said plurality of disk drives connected to a channel being connected through a single bus by means of which each disk drive is independently accessible.

Using such an arrangement, some degree of improvement in accessing speed is obtained because the bus allows each group of disks to be accessed independently, and consequently information on one disk can be accessed largely simultaneously with the information on another group, bearing in mind that the major proportion of the time taken to access a given disk is the time taken for the read/write head of the disk to seek and find the

appropriate sector of the disk, and for the disk to rotate to the correct angular position.

Preferably, however, the computer storage system incorporates the use of a segmented buffer as hereinbefore described together with the array of magnetic disk drives organised in RAID-3 format. In such an arrangement the multiple accessibility of the data stored in the memory is enhanced to its greatest potential.

Preferably, up to seven disk drives are employed on each of five channels, and thus the overall data storage capacity of the system is expanded by sevenfold.

Thus such an array, particularly according to both aspects of the present invention, provides large scale storage of information together with the faster data transfer rates and better performance with regard to multi-user applications, and security in the event of any one drive failure (per group). Indeed, the mean time between failures (MTBF) of such an array (when meaning the mean time between two simultaneous drive failures (per group), and which is required in order to result in information being lost beyond recall) is measured in many thousands of years with presently available disk drives each having individual MTBFs of many thousands of hours.

Examples of the present invention will now be described with reference to the accompanying drawings in which:

Figure 1 is a block diagram of the controller architecture of a disk array system according to one embodiment of the present invention.

Figure 2 illustrates the operation of the data splitting hardware.

Figure 3 illustrates the read/write data cell matrix.

Figure 4 illustrates a write data cell.

Figure 5 illustrates a read data cell.

| | |
|---|---|
| Figure 6 | is a flow diagram illustrating the software steps in write operations |
| Figure 7 | is a flow diagram illustrating the software steps in read operations |
| Figures 8 and 9 | are flow diagrams illustrating the software steps for read ahead and write behind |
| Figure 10 | is a flow diagram illustrating the software steps involved to restart suspended transfers |
| Figure 11 | is a flow diagram illustrating the software steps involved in cleaning up segments |
| Figures 12 and 13 | are flow diagrams illustrating the steps involved for input/output control. |

Figure 1 illustrates the architecture of the raid 35 disk array controller.

In Figure 1 of the drawings the internal interface of the computer memory controller 10 is termed the ESP data bus interface and the interface to the host computer is termed the SCSI interface. These are provided in interface 12. The SCSI bus interface communicates with the host computer (not shown) and the ESP interface communicates with a high performance direct memory access (DMA) unit 14 in a host interface section 11 of the computer memory controller 10. The ESP interface is 16 bits (one word) wide.

The host interface section communicates with a central buffer management (CBM) section 20 which comprises a central controller 22, in the form of a suitable microprocessor such as the Intel 80376 Microprocessor, and data splitting and parity control (DSPC) logic circuit 24. These perform the function of splitting information received from the host computer into four channels, and generating parity information for the fifth channel. The

DSPC 24 also combines the information on the first four channels and, after checking against the parity channel, transmits the combined information to the host computer. Furthermore, the DSPC 24 is able to reconstruct the information from any one channel, should that be necessary, on the basis of the information from the other four channels.

The DSPC 24 is connected to a central buffer 26 which is divided into five channels A to E, each of which is divisible into buffer segments 28. Each central buffer channel 26,A through 26,E have the capacity to store up to half a megabyte of data for example, depending on the application required. Each segment may be as small as 128 kilobytes for example so that up to 16 segments can be formed in the buffer.

The central buffer 26 communicates with five slave bus controls 32 under the direction of a slave bus controller 34 in a slave bus interface (SBI) section 30 of the memory controller 10. The slave bus controller 34 operates under the direction of the central controller 22.

Each slave bus controller 32,A through 32,E communicates with up to seven disk drives 42,0 to 42,6 along SCSI buses 44,A through 44,E so that the drives 42,0,A through 42,O,E form a bank, 0 of five disk drives and so also do drives 42,1,A through 42,1,E etc. to 42,6,A through 42,6,E. The seven banks of five drives effectively each constitute a single disk drive, each individually and independently accessible. This is made possible by the use of SCSI buses, which allow for eight device addresses. One address is taken up by the slave bus control 32 whilst the seven remaining addresses are available for seven disk drives. The storage capacity of each channel can therefore be increased sevenfold and the slave bus controller 32 is able to access any one of the disk drives 42 in the channel independently.

This arrangement of banks of disk drives is not only applicable to the arrangement shown in Figure 1, but is also applicable to the RAID-3 arrangement. Information stored in the disk drives of one bank can be accessed virtually simultaneously with information being accessed from the disk drives of another bank. This arrangement therefore gives an enhancement in access speed to data stored in an array of disk drives. No enhancement of speed would of course occur where information requested from two applications is stored in the same bank of disks. However, in theory at least the chance of two simultaneous requests for information being found in the same bank is 1/n where n is the number of banks employed. This is taken care of by the I/O software.

In so far as the host computer is concerned, its memory 10 consists of a number of sectors each identified by a unique address number. Where or how these sectors are stored on the various disk drives of the memory 40 is a matter of no concern to the host computer, it must merely remember the address of the data sectors it requires. Of course, addresses themselves may form part of the data stored in the memory.

On the other hand, one of the functions of the central controller 22 is to store data on the various disk drives efficiently. Moreover each sector in so far as the host is concerned, is split between four disk drives in the known RAID-3 format. The central controller 22 arranges to store sectors of information passed to it by the host computer, in an ordered fashion so that a sector on any given disk drive is likely to contain information which logically follows from a previous adjacent sector.

When the host computer requires data, the read request is received by the central controller 22 which passes the request to the slave bus interface (SBI) controller 34. The SBI controller 34 instructs the slave bus control 32 to read the disk banks 40 and select the appropriate data from the appropriate banks of disks. The DSPC circuit 24 receives the requested data and checks it is accurate against the check data in channel E.

If there is any error detected by the parity check, the faulty drive is isolated and the system arranged to continue working employing the four good channels, in the same way and with no loss of performance, until the faulty drive is replaced and rebuilt with the appropriate information.

Assuming however that the data is good, the central controller 22 first responds to the data read request by transferring the information to the SCSI interface 12. However, it also instructs further information logically sequential to the requested information to be read. This is termed "read ahead information". Read ahead information up to the capacity presently allocated by the central controller 22 to any one of the data buffer segments 28 is then stored in one buffer segment 28.

When the host computer makes a further request for information, it is likely that the information requested will follow on from the information previously requested. Consequently, when the central controller 22 receives a read request, it first interrogates those buffer segments 28 to determine if the required information is already in the buffer. If the information is there, then the central controller 22 can respond to the user request immediately, without having to read the disk drives. This is obviously a much faster procedure and avoids the seek delay.

On those occasions when the required information is not already in the buffer, then a new read of the disk drives is required. Again, the requested information is passed on and sequential read ahead information is fed to another buffer segment. This process continues until all the buffer segments are filled and the system is maintained with its segments permanently filled. Of course, there comes a point when all the segments are filled, but still the disk drives must be read. It is only at this point that a buffer segment is finally deallocated by the central controller 22, by keeping note of which buffer segments buffers 28 are or have been used most frequently, and dumping the most infrequently used one.

During the normal busy operation of the host computer, the central controller 22 will have allocated at least as many buffer segments 28 as there are application programs, up to the maximum number of segments available. Each buffer segment will be kept full by the central controller 22 ordering the disk drive seek commands in the most efficient manner, only over-riding that ordering when a buffer segment has been, say 50% emptied by host requests or when a host request cannot be satisfied from existing buffer segments 28. Thus all buffer segments are kept as full as possible with read ahead data.

To write information to the disk drives, a similar procedure is followed. When a write instruction is received by the central controller 22 information is split by DSPC circuits 24 and appropriate check information created. The five resulting components are placed in allocated write buffer segments. The number of write buffer segments may be preselected, or may be dynamically allocated as and when required. In any event, write buffer segments are protected against de-allocating until its

information has been written to disk. Actual writing to disk is only effected under instruction from the host computer, if and when a segment becomes full and the system cannot wait any longer, or, more likely, when the system is idle and not performing any read operations.

In any event, simultaneous writes appear to be happening in so far as the host computer is concerned, because the central controller 22 is capable of handling commands very rapidly and storing writes in buffers while waiting for an opportunity for the more time consuming actual writing to disk drives.

This does not mean however, that in the event of power failure, some writes, which the user will think have been recorded on disk, may in fact have been lost by virtue of its temporary location in the random access buffer at the time of power failure. In that event a restored disk drive system from back-up copies is required.

Alternatively, a hardware switch can be provided to ensure that all write instructions are effected immediately, with write information only being stored in the buffer segments transiently before being written to disk. This removes the fear that a power loss might result in data being lost which was thought to have been written to disk although not actually effected by the memory system. There is still however, the unlikely exception that information may be lost when a power loss occurs very shortly after a user has sent a write command, but in that event, the user is likely to be conscious of the problem. If this alternative is utilised, it does of course affect the performance of the computer.

The detailed operation of the hardware data splitting, parity generation and checking logic, and buffer interface logic will now be described with reference to Figures 2 to 5.

Referring to Figure 2, the controllers internal interface to the host system hardware interface is 16 bits (one word) wide. This is the ESP data bus. For every four words of sequential host data, one 64 bit wide slice of internal buffer data is formed. At the same time, an additional word or 16 bits of parity data is formed by the controller; one parity bit for four host data bits. Thus the internal width of the controller's central data bus is 80 bits. This is made up of 64 bits of host data and 16 bits of parity data.

The data splitting and parity logic 24 is split up into 16 identical read/write data cells within the customised ASICS (application specific integrated circuits) design of the controller. The matrix of these data cells are shown in Figure 3. Each of these data cells handles the same data bit from the ESP bus for the complete sequence of four ESP 16 bit data words. That is, with reference to Figure 2, each data cell handles the same bit from each ESP bus word 0,1,2 and 3. At the same time, each data cell generates/reads the associated parity bit for these four 16 bit ESP bus data words.

For explanation purposes, only the first data bit 0 (DB0) will be described. Data bits DB1 through DB15 will be identical in operation and description.

Four basic operations are performed, namely

1. Writing host data
2. Reading of data to the host
3. Regeneration of "single failed channel" data during host read operations.
4. Rebuilding of data on a failed disk drive unit.

Writing of host data to the disk drive array

Referring now to Figure 4, as the corresponding data bit from each host 16 bit word is received on the ESP data bus, each of these four bits is temporarily stored/latched in devices G38 through G41. As each bit appears on the ESP bus, it is steered through the multiplexor under the control of the two select lines to the relevant D-type latches G33 through G36, commencing with G33. At the end of this initial operation, the four host 16 bit words (64 data bits) will have been stored in the relevant gates G38 through G41 within all 16 data cells. The four DB0 data bits are now called DB0-A through DB0-D.

During the write operations, the RMW (buffer read modify write) control signal is set to select input A from all devices G38 through G42. Under these situations, the rebuild line is not used (don't care).

As each bit is clocked into the data cell, the corresponding parity data bit is generated via G31, G32, and G37. At the end of the sequence of the four bit 0's from each of the four incoming ESP bus host data words, the resultant parity bit will have been generated and stored on device G42. This is accomplished as follows. As the first bit-0 (DB0-A) appears on the signal DB0, the INIT line is driven high/true and the output from the gate G31 is driven low/off. Whatever value is present on DB0 will appear on the output of gate G32, and at the correct time will be clocked into the D-type G37. The value of DB0 will now appear on the Q output of G37. The INIT signal will now be driven low/off, and will now aid the flow of data through G31 for the next incoming three data bits on DB0. Whatever

value was stored as DB0-A on the output of gate G37 will now appear on the output of gate G31, and as the second DB0 bit (DB0-B) appears on the signal DB0, an Exclusive OR value of these two bits will appear on the output of gate G32. At the appropriate time, this new value will be clocked into the device G37. At the end of the clock cycle, the resultant Q output of G37 will now be the Exclusive OR function of DB0-A and DB0-B. This value will now be stored on device G42. The above operation will continue as the remaining two DB0 bits (DB0-C and DB0-D) appear on the signal DB0. At the end of this operation, the accumulative Exclusive OR function of all bits DB0-A through DB0-D will be stored on device G42, and at the same time, bits DB0-A through DB0-D will be stored on devices G38 through G41 respectively.

The accumulative Exclusive OR (XOR) value of DB0-A through DB0-D is generated in this manner so as to preserve buffer timing and synchronisation procedures.

The five outputs DB0-A through DB0-E are present for all data bits 0 through 15 of the four host data words. The total of 80 bits are now stored in the central buffer memory (DRAM). The whole procedure is repeated for each sequence of four host data words (8 host data bytes).

As each "sector" of slave disk drive data is assembled in the central buffer, it is written to all slave disk drives (to channel A through channel E) within the same bank of disk drives.

If a failed slave channel, or disk drive exists, then the controller will mask out that drive's data and no data will be written to that channel/disk drive. However, the data will be assembled in the central buffer in the normal manner.

Reading of array disk drive data to the host system

Referring now to Figure 5, in response to a host request, data is read from the disk array and placed in the central buffer memory 26. Also, in the reverse procedure to that for write operations, the 80 bits of central buffer data are loaded into devices G10 through G14 for each bit (4 data bits and 1 parity bit). Again we will only consider DB0. The resulting five bits are DB0-A through DB0-E. All read operations are checked for correct parity by regenerating a new parity bit and comparing this bit with the bit read from the slave disk drives.

Initially, the case of a fully functioning array will be considered with no faulty slave disk drives. In this case all mask bits (mask-A through mask-E) will be low/false, and all bits from the central buffer 26 will appear on the outputs of devices G10 through G14 via "A" inputs. Also, all data bits will appear on the outputs of devices G6 through G9 via their "A" inputs. After the central buffer read operation, the four data bits will simultaneously appear on the outputs of devices G6 through G9. In the reverse procedure to that for write operations, all data bits DB0-A through DB0-D will be reassembled on the ESP data bus through the mutilplexor under the control of the two select lines. As the data bits are read from the central buffer 26, the parity data bit is regenerated by the Exclusive OR gate G4 and compared to gate G2 with the parity data read from the slave disk drives at device G14. If a difference is detected, a NMI "non-maskable interrupt" is generated to the master processor device via gate G3. All read operations will terminate immediately.

Gate G5 suppresses the effect of the parity bit DB0-E from the generation of the new parity bit. Gate G1 will suppress NMI operations if any slave disk drive has failed and the resultant mask bit has been set high/true.

Also, gate G1, in conjunction with gate G5, will allow the read parity bit DB0-E to be utilised in the regeneration process at gate G4, should any channel have failed.

Regeneration of "single failed channel" data during host read operations

Referring to Figure 5, the single failed disk drive/channel will have its mask bit set high/true under the direction of the controller software. The relevant gates within G6 through G9 and G10 through G14 for the failed channel/drives will have their outputs determined by their "B" inputs, not their "A" inputs. Also, G1 will suppress all NMI generation, and together with gate G5, will allow parity bit DB0-E to be utilised at gate G4. In this situation, the four valid bits from gates G10 through G14 will "regenerate" the "missing" data at gate G4, and the output with gate G4 will be fed to the correct ESP bus data bit DB0 via a "B" input at the relevant gate G6 through G9.

For example consider the channel 2 disk drive to be faulty, and mask bit mask-C will be driven high/true. The output of gate G12 will be driven low and will not contribute to the output of gate G4. Also, the output of gate G1 will be driven low/false and will both suppress NMIs, and will allow signal DB0-E to be fed by gate G5 to gate G4. Gate G4 will have all correct inputs from which to regenerate the missing data and feed the data to the output of device G8 via its "B" input. At the correct time, this bit will be fed through the multiplexor to DB0.

Rebuilding of data on a failed disk drive unit

Referring now to Figures 4 and 5, to rebuild data, the memory controller must first read the data from the functioning four disk drives, regenerate the missing

drive's data, and finally write the data to the failed disk drive after it has been replaced with a new disk drive.

With reference to Figure 5 and the example given above for "regeneration of single failed channel data during host read operations", under rebuild conditions the outputs from gates G6 through G9 will not be fed to the ESP data bus. However, the regenerated data at the output of gate G4 will be fed to the "B" inputs of gates G38 through G42 of the write data cell in Figure 4. Under rebuild conditions, the RMW signal will be set high/true and the outputs of devices G38 through G42 will be determined by the value of the rebuild data on signal rebuild.

All channels of the central buffer memory 26 will have their data set to the regenerated data, but only the single replaced channel data will be written to the new disk drive under software control.

Detection of faulty channel/disk drive

The detection of a faulty channel/slave disk drive is as per the following three main criteria:-

1. The master 80376 processor detects an 80186 channel (array controller electronics) failure due to an "interprocessor" command protocol failure.
2. An 80186 processor detects a disk drive problem i.e. a SCSI bus protocol violation.
3. An 80186 processor detects a SCSI bus hardware error. This is a complete channel failure situation, not just a single disk drive on that SCSI bus.

After detection of the fault condition, the channel/drive "masking" function is performed by the master 80376 microprocessor.

Under fault conditions, the masked out channel/drive is not written to or read from by the associated 80186 channel processor.

Figure 6 through to 13 are diagrams illustrating the operation of the software run by the central controller 22.

Figure 6 illustrates the steps undertaken during the writing of data to the banks of disk drives. Initially the software is operating in "background" mode and is awaiting instructions. Once an instruction from the host is received indicating that data is to be sent, it is determined whether this is sequential within an existing segment. If data is sequential then this data is stored in the segment to form sequential data. If no sequential data exists in a buffer segment then either a new segment is opened (the write behind procedure illustrated in Figure 8) and data is accepted from the host, or the data is accepted into a transit buffer and queued ready to write into a segment. If there is no room for a new segment then the segment is found which has been idle for the most time. If there are no such segments then the host write request is entered into a suspended request list. If a segment is available it is determined whether this is a read or write segment. If it is a write segment then if it is empty it is de-allocated. If it is not empty then the segment is removed from consideration for de-allocation. If the segment is a read segment then the segment is de-allocated and opened ready to accept the host data.

The write behind procedure is illustrated in Figure 8 and if there are any write segments open which need to be emptied, then a write request is queued for the I/O handler for each open segment with data in it.

Figure 7 illustrates the steps undertaken during read operations. Initially, the controller is in a "background" mode. When a request for data is received

from the host computer, if the start of the data requested is already in a read segment then data can be transferred from the central buffer 26 to the host computer. If the data is not already in the central buffer 26, then it is ascertained whether it is acceptable to read ahead information. If it is not acceptable then a read request is queued. If data is to be read ahead then it is determined whether there is room for a new segment. If there is then a new segment is opened and data is read from the drives to the buffer segment and is then transferred to the host computer. If there is no room for a new segment then the segment is found for which the largest time has elapsed since it was last accessed, and this segment is de-allocated and opened to accept the data read from the disk drives.

In order to keep the buffer segments 28 full, the read ahead procedure illustrated in Figure 9 is formed. It is determined whether there are any read segments open which require a data refresh. If there is such a segment then a read request for the I/O handler for the segment is queued.

Figure 10 illustrates the software steps undertaken to restart suspended transfers. It is first determined whether there are suspended host write requests in the list. If there is it is determined whether there is room for allocation of a segment for suspended host write requests. A new segment for the host transfer is opened and the host request which has been suspended longest is determined and data is accepted from the host computer into the buffer segment.

Figure 11 illustrates a form of "housekeeping" undertaken by the software in order to clean up the segments in the central buffer 26. It is determined at a point that it is time to clean up the buffer segments. All

the read segments which have times since the last access time larger than a predetermined limit termed the "geriatric limit" are found and reallocated. Also it is determined whether there are any such write segments and if so write operations are tidied up.

Figure 12 illustrates the operation of the input/output handler, whilst Figure 13 illustrates the operation of the input/output sub system.

All these procedures are performed by software which may be run on the central (80376) controller 22 in order to control and efficiently manage the transfer of data in the buffer segments 28, in order that the buffer 26 is kept as full as possible with data sequential to data requested by the host computer.

Sector Translation

A problem has been experienced with the disk drives available to form the slave disk drive banks 40. As mentioned above host data arriving in "sectors" is split into four. This arrangement relies upon the slave disk drives of the array being able to be formatted with sector sizes exactly one quarter of that used by the host. A current standard sector size is 512 bytes, with a resultant slave disk sector size requirement of 128 bytes.

Until recently this has not been a problem, but due to the speed and complexity of electronics, disk drives above the 500 megabyte level can typically only be formatted to a minimum of 256 bytes per sector. Further, new disk drives above the 1 gigabyte capacity, can typically only support a minimum of 512 byte sectors. This would mean that the controller would only be able to support host sector sizes of two kilobytes.

This problem has been overcome by applying a technique termed "sector translation". In this technique each slave disk sector contains four host sectors in what is termed "virtual" slave sectors of 128 bytes. In this technique if the host requires a single sector of 512 bytes, then the controller has to extract an individual sector of 128 bytes from within the larger actual 512 bytes slave disk drive sector. When writing data, for individual writes of a single sector, or less than four correctly grouped sectors, the controller has first to read the required overall sector, then modify the data for the actual part of the sector that is necessary, and then finally write the overall slave disk sector back to the disk drive. This is a form of read modify write operation and can slow down the transfer of data to the disk drives considerably.

However, for large transfers of data to or from the disk drives, the affect of this problem is minimal and is not noticed by the host computer. For random access of small amounts of data, the RAID-3 controller is inferior to the RAID-5 controller.

From the embodiments hereinabove described it can be seen that the controller of the present invention provides for large scale sequential data transfers from memory units for multi-users of a host computer.

The present invention is applicable to any standard host interface or slave interface and is not limited to the use of an SCSI bus as shown in Figure 1.

While the invention has been described with reference to specific elements and combinations of elements, it is envisaged that each element may be combined with other or any combination of other elements. It is not intended to limit the invention to the particular

combinations of elements suggested. Furthermore, the foregoing description is not intended to suggest that any element mentioned is indispensable to the invention, or that alternatives may not be employed. What is defined as invention should not be construed as limiting the extent of the disclosure of this specification.

CLAIMS

1. A computer memory controller for interfacing to a host computer comprising a buffer means for interfacing to at least one memory unit and for holding data read thereto or therefrom; said buffer means controlled to form a plurality of buffer segments for addressably storing data requested by said host computer and further data which is logically sequential thereto; and control means operative to control the transfer of data to said host computer in response to requests therefrom by first addressing said buffer segments to establish whether the requested data is contained therein and if so supplying said data to said host computer, and if the requested data is not contained in the buffer segments reading said data from the or each memory unit, supplying said data to said host computer, reading from the or each memory unit further data which is logically sequential to the data requested by said host computer and storing said further data in a buffer segment; said control means controlling the buffer means to control the number and size of said buffer segments.

2. A computer memory controller as claimed in Claim 1 for use with at least three memory units, including a logic circuit connected to said buffer means to recombine bytes or groups of bits successively read from successive ones of a group of said memory units and stored in said buffer segments, said logic circuit including parity means operative to use a check byte or groups of bits read from one of said memory units to regenerate data read from said group of memory units if one of said group of memory units fails, said buffer means being divided into a number of channels corresponding to the number of memory units, each channel being divided into associated portions of buffer segments.

3. A computer memory controller as claimed in Claim 2, wherein said logic circuit is operative to split data input from said host computer such that successive bytes or groups of bits from said host computer are temporarily stored in said buffer segments before being successively applied to successive ones of a group of said memory units, and said parity means is operative to generate a check byte or group of bits from said data for temporary storage in a buffer segment before being stored in at least one said memory unit

4. A computer memory controller as claimed in any preceding claim, wherein said control means is operative to reduce the size of existing buffer segments on each occasion that a request for data from said host computer cannot be complied with from the further data stored in existing ones of said buffer segments, dynamically allocate a new segment of said buffer means for further data to the data requested, and continue this process until the size of each buffer segment in some predetermined minimum, whereupon, at the next request for data not available in a buffer segment, the buffer segment least frequently utilised is employed.

5. A computer memory controller as claimed in claim 4 wherein said predetermined minimum size of said buffer segments is 128 kilobytes.

6. A computer memory controller as claimed in any preceding claim wherein a number of said buffer segments are assigned for storing data to be written to the or each memory unit.

7. A computer memory controller as claimed in any preceding claim, wherein said buffer means is adapted for interfacing to five memory units, one said memory unit holding said check bytes or groups of bits.

8. A computer memory controller as claimed in any preceding claim, wherein said buffer is adapted for interfacing to disk drives.

9. A method of controlling an array of memory units for use with a host computer comprising the steps of receiving from said host computer a read request for data stored on the memory units, checking a plurality of buffer segments to establish whether the requested data is in said buffer segments, either complying with said request by transferring the data in said buffer segments to said host computer, or first reading said data from said memory units into one buffer segment and then complying with said request, reading from the memory units further data logically sequential to the data requested and storing said data in said buffer segment.

10. A method as claimed in Claim 9, including the steps of recombining bytes or groups of bits successively read from successive ones of a group of said memory units and stored in said buffer segments, wherein a check byte or group of bits read from one of said memory units is used to regenerate data read from said group of memory units if one of said group of memory units fails.

11. A method as claimed in Claim 9 or Claim 10, including the steps of receiving data from said host computer, splitting said data such that successive bytes or groups of bits are temporarily stored in successive portions of said buffer segments before being successively applied to successive ones of a group of said memory units, and generating a check byte or group of bits from said data for temporary storage in a buffer segment before being stored in at least one said memory unit.

12. A method as claimed in any of Claims 9 to 11, including the steps of reducing the size of existing buffer segments on each occasion that a request for data from said host computer cannot be complied with from the further data stored in existing ones of said buffer segments, dynamically allocating a new segment of said buffer for further data to the data requested, and continuing this process until the size of each buffer segment is some predetermined minimum, whereupon at the next request for data not available in a buffer segment the buffer segment least frequently utilised is employed.

13. A computer memory controller for a host computer comprising buffer means for interfacing to at least three memory units arranged in parallel and for holding information read from said memory units; a logic circuit connected to said buffer means to recombine bytes or groups of bits successively read from successive ones of a group of said memory units; parity means operative to use a check byte or group of bits read from one of said memory units to regenerate information read from said groups of memory units if one of said group of memory units fails; said buffer means being controlled to form a number of buffer

segments each storing data requested by an application run on said host computer and further data which is logically sequential thereto, and a controller for controlling the transfer of data to said host computer in response to requests from said host computer by checking said buffer segments to establish whether the requested data is in said buffer segment and supplying said data to said host computer, or reading said data from said memory units, supplying said data to said host computer, reading from said memory units further data which is logically sequential to the data requested by said host computer and storing said further data in a buffer segment.

14. A computer memory controller for a host computer comprising buffer means for interfacing at least three memory units arranged in parallel, a logic circuit connected to said buffer means to split data input from said host computer such that successive bytes or groups of bits from said host computer are temporarily stored in said buffer means before being successively applied to successive ones of a group of said memory units, said logic circuit being further operative to recombine bytes or groups of bits successively read from successive ones of said group of said memory units into said buffer means, said logic circuit including parity means operative to generate a check byte or group of bits from said data for temporary storage in said buffer means before being stored in at least one said memory unit, and operative to use said check byte to regenerate said data read from said group of memory units if one of said group of memory units fails, said buffer means being divided into a number of channels corresponding to the number of memory units, each said channel being divided into associated portions of segment

buffers, buffer segments containing successive bytes or groups of bits corresponding to data for an application being run by said host computer, and control means operative to control the transfer of data and check bytes or groups of bits to and from said memory units in response to commands from said host computer, and operative to control the number and size of said buffer segments.

15. A computer memory controller as claimed in Claim 14, wherein said control means is operative to control the transfer of data and check bytes or groups of bits to said host computer in response to request therefrom by checking said buffer segments to establish whether the requested data is contained therein and supplying said data to said host computer, or reading said data and check bytes or groups of bits from said memory units, supplying said data to said host computer, reading from said memory units further data and check bytes or groups of bits which is logically sequential to the data requested by the host computer, and storing said further data and check bytes and groups of bits in a buffer segment.

16. A computer memory controller as claimed in Claim 14 or Claim 13, wherein said control means is operative to reduce the size of existing buffer segments on each occasion that a request for data from said host computer cannot be complied with from the further data stored in existing ones of said buffer segments, dynamically allocate a new segment of said buffer for further data to the data requested, and continue this process until the size of each buffer segment is some predetermined minimum whereupon, at the next request for data not available in a buffer segment, the buffer segment least frequently utilised is employed.

17. A computer memory controller as claimed in any of Claims 14 to 16, wherein said buffer means is adapted for interfacing to five memory units, one said memory unit holding said check bytes or groups of bits.

18. A computer storage system comprising an array of magnetic disk drives organised in RAID-3 format having at least three channels, said array comprising a plurality of disk drives connected to each said channel, each of said plurality of disk drives connected to a channel being connected through a single bus by means of which, each disk drive is independently accessible.

19. A computer storage system as claimed in Claim 18, wherein said array is provided with five channels, one said channel accessing check disks.

20. A computer storage system as claimed in Claim 19, wherein each seven disk drives are connected to each channel.

21. A computer storage system comprising at least three memory units arranged in parallel, each said memory unit comprising a plurality of disk drives connected by a single bus such that each disk drive of said memory unit is independently accessible, buffer means for interfacing to said memory units and for holding information read from the memory units, a logic circuit connected to said buffer means to recombine bytes or groups of bits successively read from successive ones of a group of said memory units, parity means operative to use a check byte or group of bits read from one of said memory units to regenerate information read from said groups of memory units if one of said group of memory units fails, said buffer means being

controlled to form a number of buffer segments each storing data requested by an application run on said host computer and further data which is logically sequential thereto, and a controller for controlling the transfer of data to said host computer in response to requests from said host computer by checking said buffer segments to establish whether the requested data is in said buffer segment and supplying said data to said host computer, or reading said data from said memory units, supplying said data to said host computer, reading from said memory units further data which is logically sequential to the data requested by said host computer and storing said further data in a buffer segment.

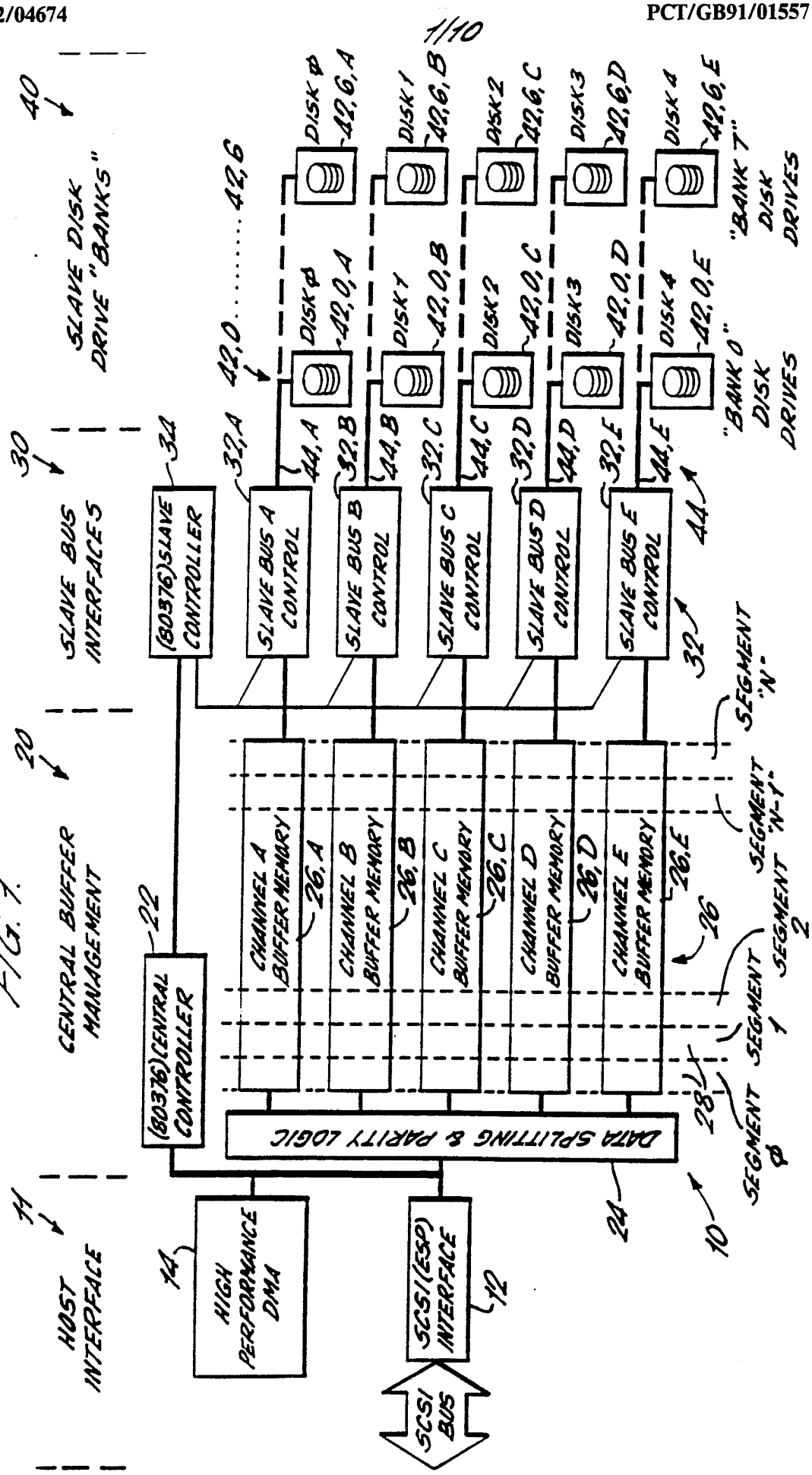

1/10
HOST INTERFACE
14
HIGH PERFORMANCE DMA
SCSI/(ESP) INTERFACE
12
SCSI BUS
CENTRAL BUFFER MANAGEMENT
20
(80376) CENTRAL CONTROLLER
22
DATA SPLITTING & PARITY LOGIC
24
10
CHANNEL A BUFFER MEMORY 26,A
CHANNEL B BUFFER MEMORY 26,B
CHANNEL C BUFFER MEMORY 26,C
CHANNEL D BUFFER MEMORY 26,D
CHANNEL E BUFFER MEMORY 26,E
26
SEGMENT "N"
SEGMENT "N-1"
SEGMENT 2
SEGMENT 1
SEGMENT 0
28
32
SLAVE BUS INTERFACES
30
(80376) SLAVE CONTROLLER
34
SLAVE BUS A CONTROL 32,A
SLAVE BUS B CONTROL 32,B
SLAVE BUS C CONTROL 32,C
SLAVE BUS D CONTROL 32,D
SLAVE BUS E CONTROL 32,E
44
44,A
44,B
44,C
44,D
44,E
SLAVE DISK DRIVE "BANKS"
40
42,0 ..... 42,6
DISK 0 42,0,A
DISK 1 42,0,B
DISK 2 42,0,C
DISK 3 42,0,D
DISK 4 42,0,E
"BANK 0" DISK DRIVES
DISK 0 42,6,A
DISK 1 42,6,B
DISK 2 42,6,C
DISK 3 42,6,D
DISK 4 42,6,E
"BANK 7" DISK DRIVES

FIG. 1.

FIG. 2.

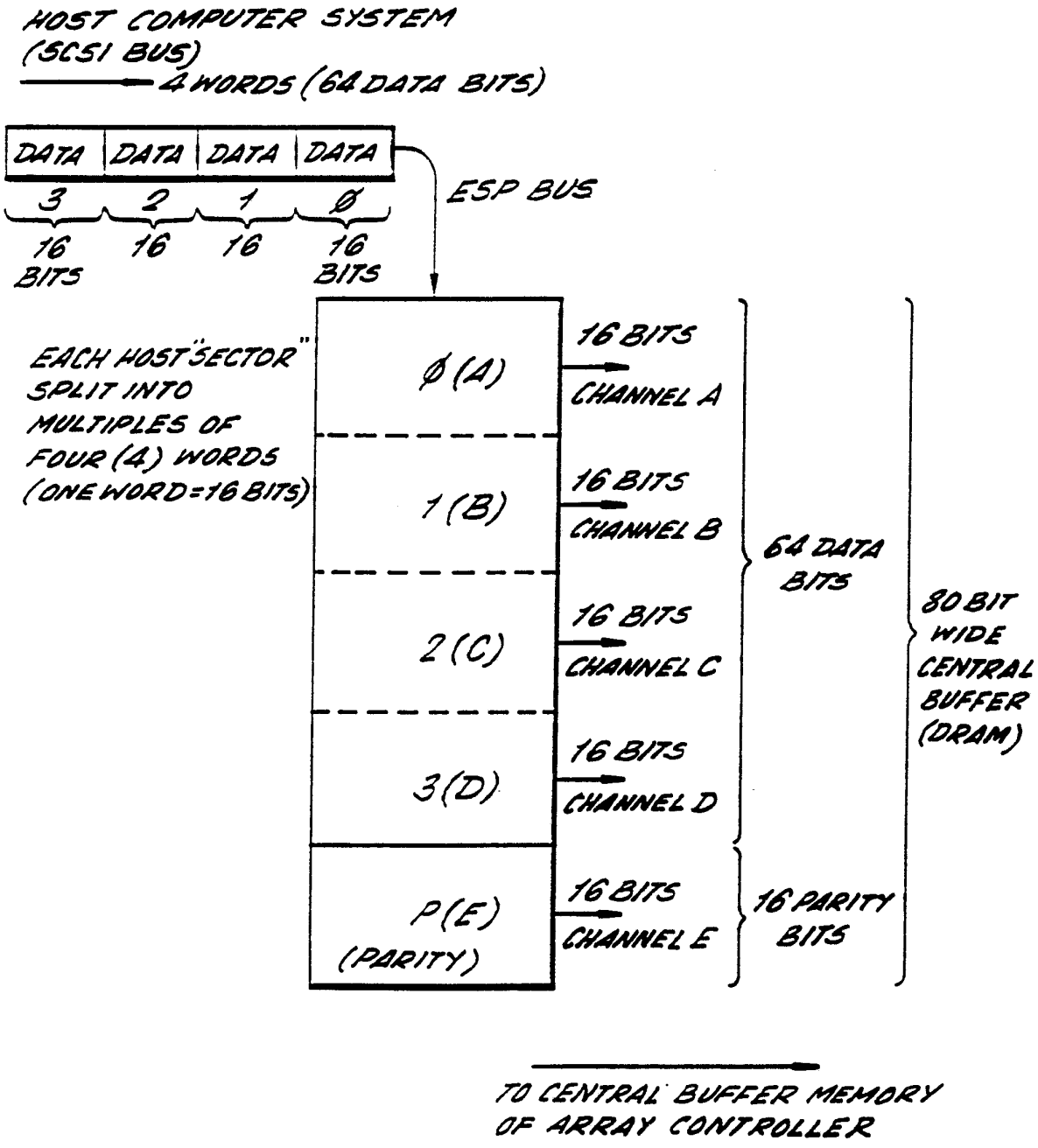

HOST COMPUTER SYSTEM
(SCSI BUS)
4 WORDS (64 DATA BITS)
DATA
DATA
DATA
DATA
3
2
1
0
16 BITS
16
16
16 BITS
ESP BUS
EACH HOST "SECTOR" SPLIT INTO MULTIPLES OF FOUR (4) WORDS (ONE WORD = 16 BITS)
0 (A)
1 (B)
2 (C)
3 (D)
P (E)
(PARITY)
16 BITS
CHANNEL A
16 BITS
CHANNEL B
16 BITS
CHANNEL C
16 BITS
CHANNEL D
16 BITS
CHANNEL E
64 DATA BITS
16 PARITY BITS
80 BIT WIDE CENTRAL BUFFER (DRAM)
TO CENTRAL BUFFER MEMORY OF ARRAY CONTROLLER

FIG. 3.

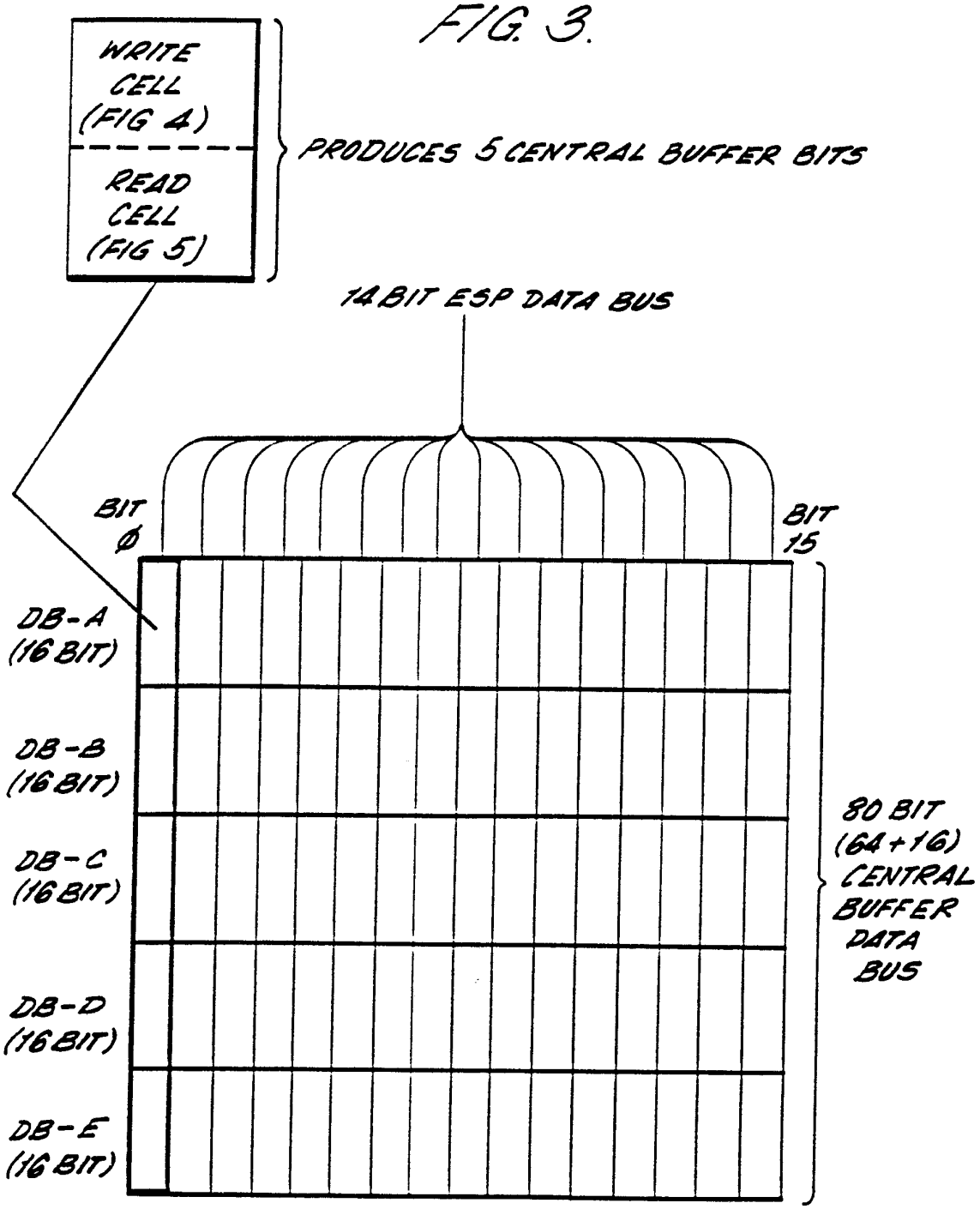
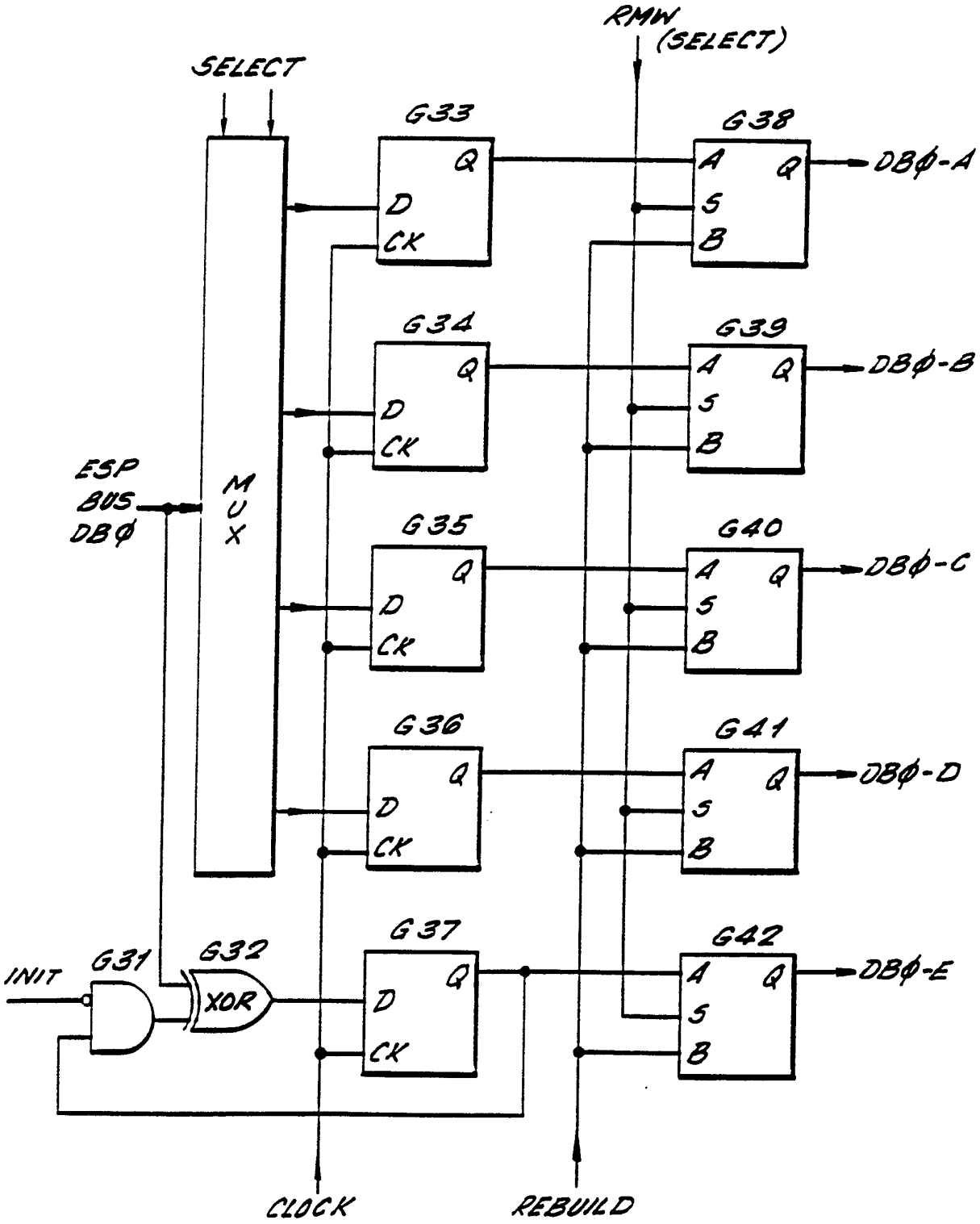

WRITE CELL (FIG 4)
READ CELL (FIG 5)
PRODUCES 5 CENTRAL BUFFER BITS
14 BIT ESP DATA BUS
BIT Ø
BIT 15
DB-A (16 BIT)
DB-B (16 BIT)
DB-C (16 BIT)
DB-D (16 BIT)
DB-E (16 BIT)
80 BIT (64+16) CENTRAL BUFFER DATA BUS

FIG. 4.


RMW
(SELECT)
SELECT
G33
G38
DBφ-A
G34
G39
DBφ-B
ESP
BUS
DBφ
MUX
G35
G40
DBφ-C
G36
G41
DBφ-D
G37
G42
DBφ-E
INIT
G31
G32
XOR
Q
D
CK
A
S
B
CLOCK
REBUILD

5/10

FIG. 5.

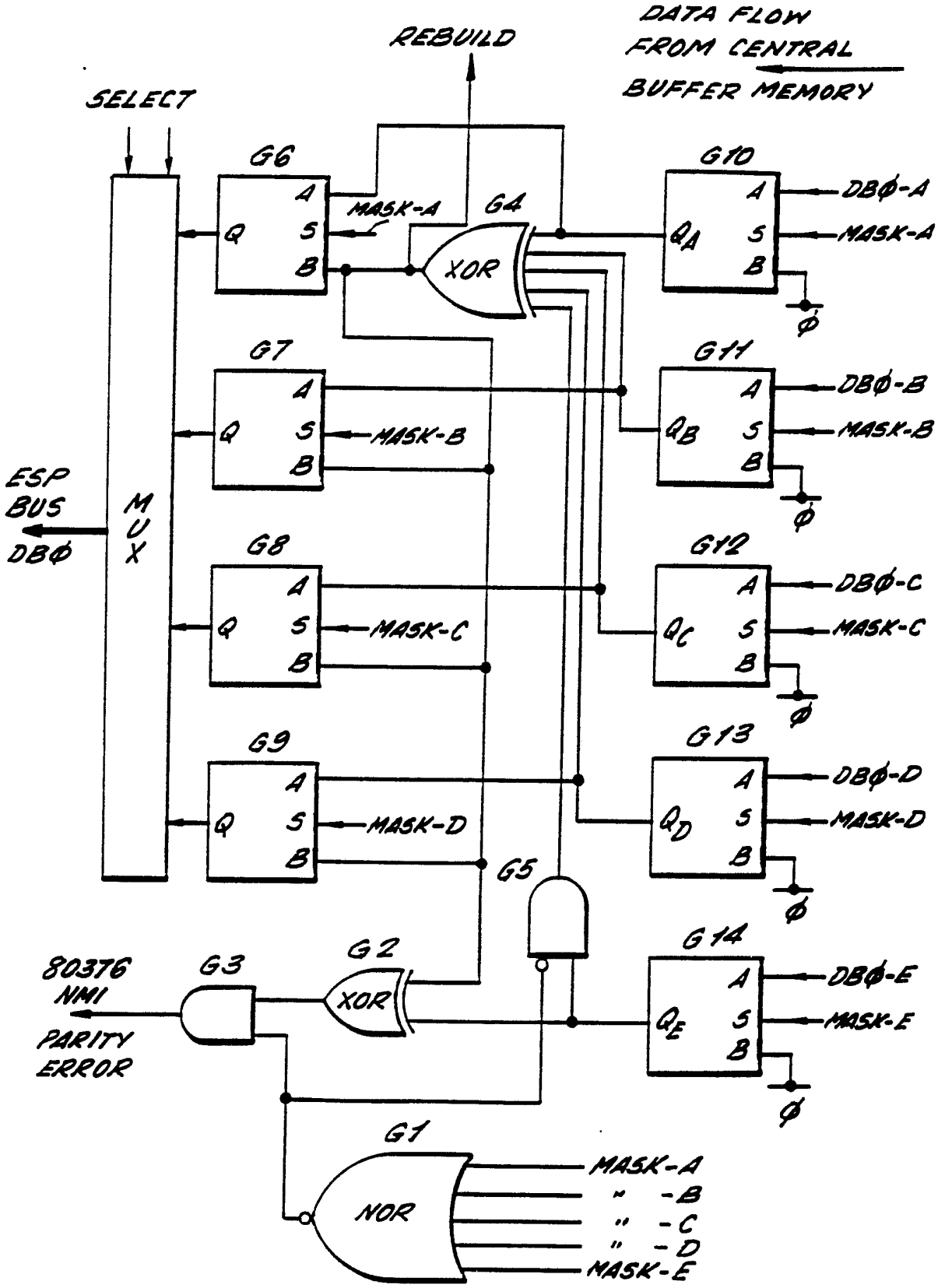

DATA FLOW FROM CENTRAL BUFFER MEMORY
REBUILD
SELECT
G6
G4
G10
MASK-A
XOR
DBφ-A
MASK-A
φ
G7
G11
MASK-B
DBφ-B
MASK-B
ESP BUS DBφ
MUX
G8
G12
MASK-C
DBφ-C
MASK-C
G9
G13
MASK-D
DBφ-D
MASK-D
G5
G14
80376 NMI
G3
G2
XOR
DBφ-E
MASK-E
PARITY ERROR
G1
NOR
MASK-A
" -B
" -C
" -D
MASK-E

FIG. 6.

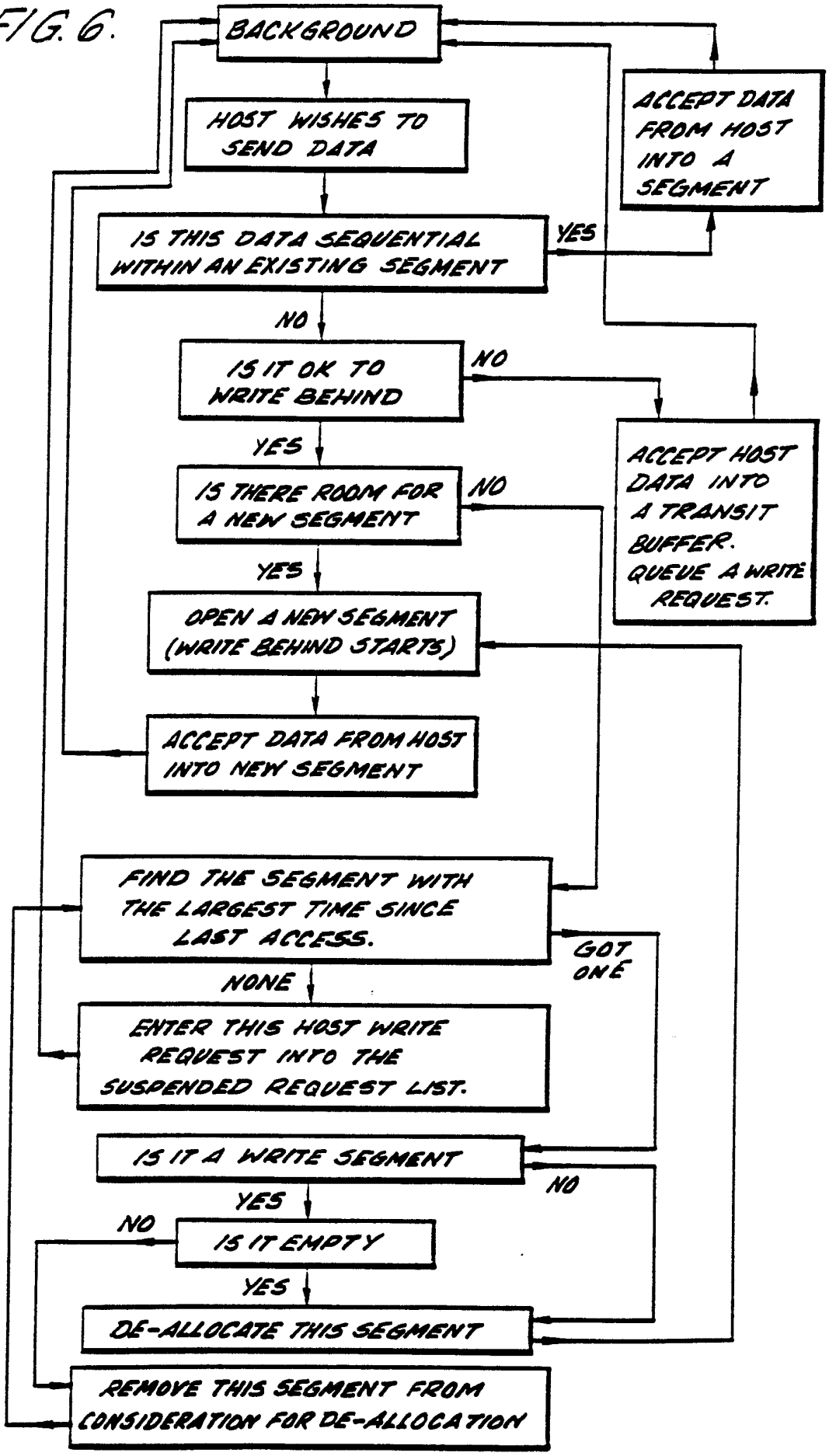

BACKGROUND
HOST WISHES TO SEND DATA
IS THIS DATA SEQUENTIAL WITHIN AN EXISTING SEGMENT
YES
ACCEPT DATA FROM HOST INTO A SEGMENT
NO
IS IT OK TO WRITE BEHIND
NO
YES
ACCEPT HOST DATA INTO A TRANSIT BUFFER. QUEUE A WRITE REQUEST.
IS THERE ROOM FOR A NEW SEGMENT
NO
YES
OPEN A NEW SEGMENT (WRITE BEHIND STARTS)
ACCEPT DATA FROM HOST INTO NEW SEGMENT
FIND THE SEGMENT WITH THE LARGEST TIME SINCE LAST ACCESS.
GOT ONE
NONE
ENTER THIS HOST WRITE REQUEST INTO THE SUSPENDED REQUEST LIST.
IS IT A WRITE SEGMENT
NO
YES
NO
IS IT EMPTY
YES
DE-ALLOCATE THIS SEGMENT
REMOVE THIS SEGMENT FROM CONSIDERATION FOR DE-ALLOCATION

FIG. 7.

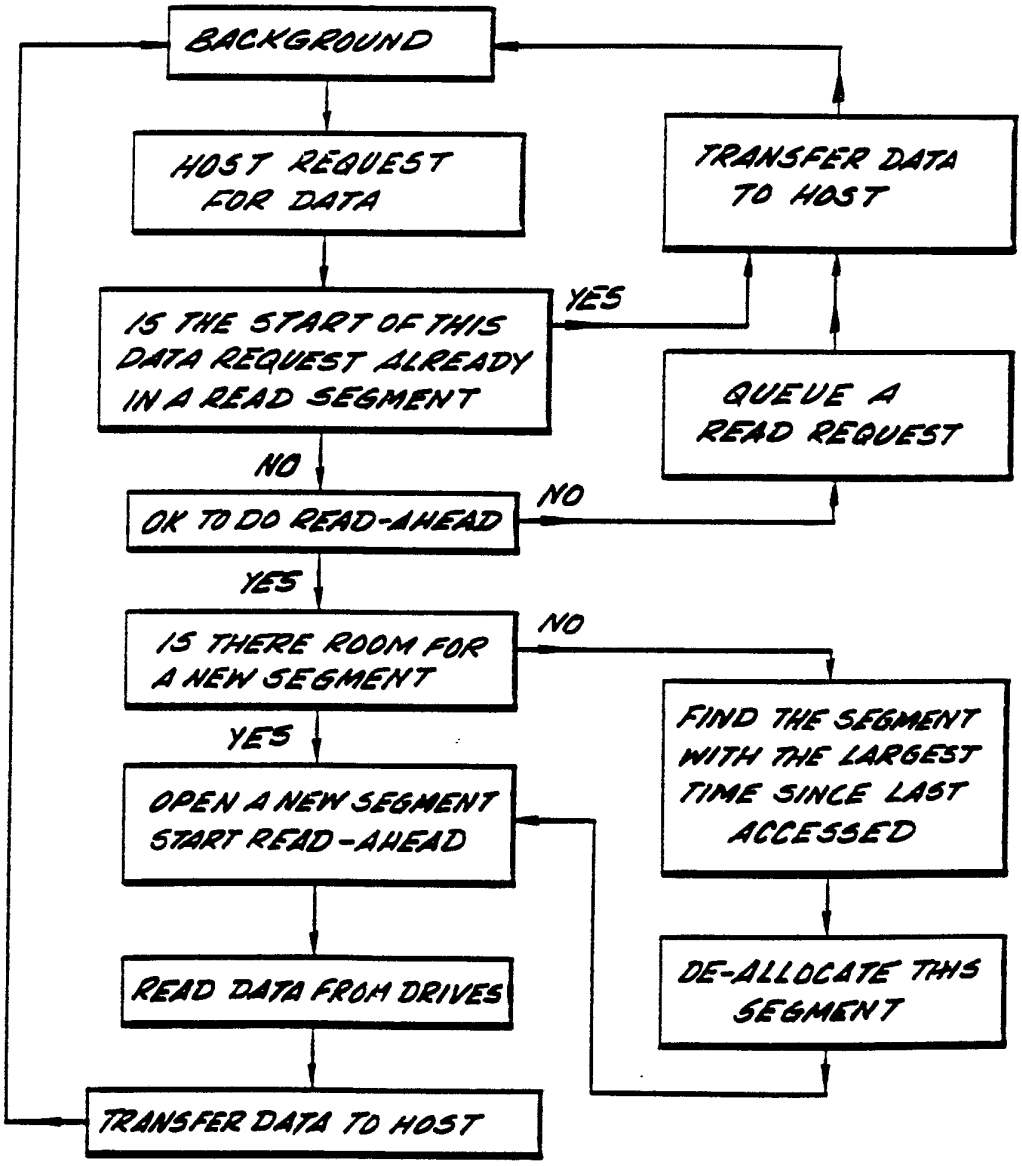

BACKGROUND
HOST REQUEST FOR DATA
IS THE START OF THIS DATA REQUEST ALREADY IN A READ SEGMENT
YES
NO
OK TO DO READ-AHEAD
NO
YES
IS THERE ROOM FOR A NEW SEGMENT
NO
YES
OPEN A NEW SEGMENT START READ-AHEAD
READ DATA FROM DRIVES
TRANSFER DATA TO HOST
TRANSFER DATA TO HOST
QUEUE A READ REQUEST
FIND THE SEGMENT WITH THE LARGEST TIME SINCE LAST ACCESSED
DE-ALLOCATE THIS SEGMENT

FIG. 8.

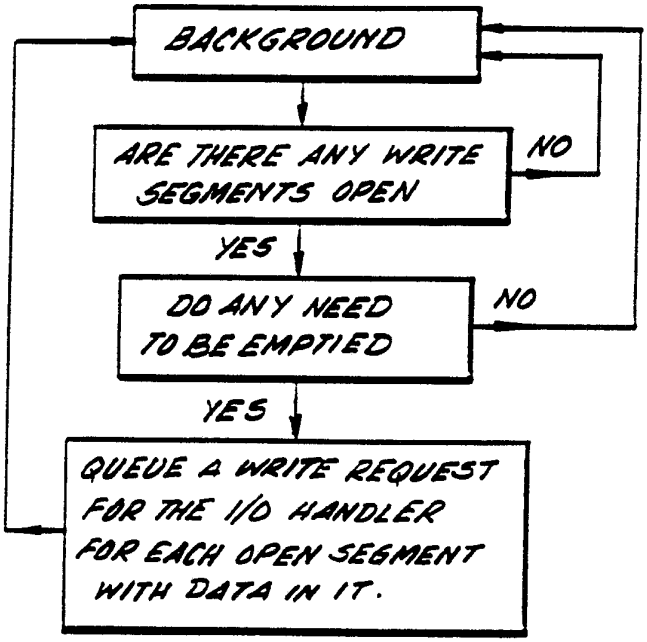

BACKGROUND
ARE THERE ANY WRITE SEGMENTS OPEN
NO
YES
DO ANY NEED TO BE EMPTIED
NO
YES
QUEUE A WRITE REQUEST FOR THE I/O HANDLER FOR EACH OPEN SEGMENT WITH DATA IN IT.

FIG. 9.

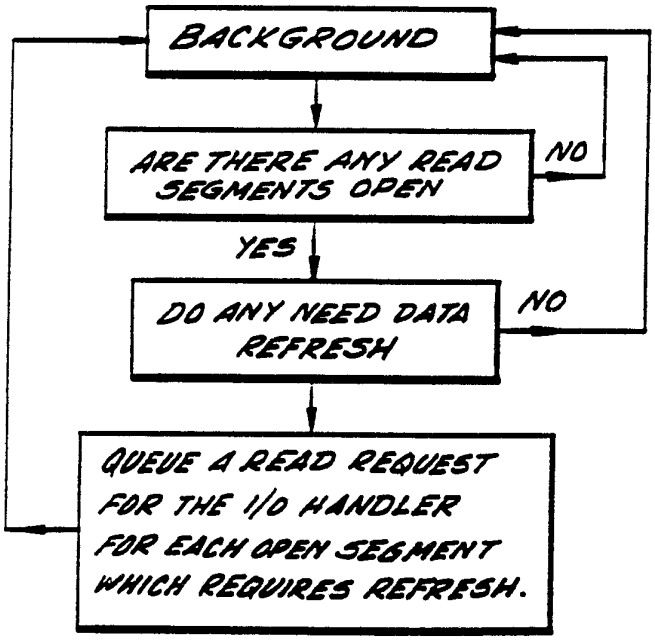

BACKGROUND
ARE THERE ANY READ SEGMENTS OPEN
NO
YES
DO ANY NEED DATA REFRESH
NO
QUEUE A READ REQUEST FOR THE I/O HANDLER FOR EACH OPEN SEGMENT WHICH REQUIRES REFRESH.

FIG. 10.

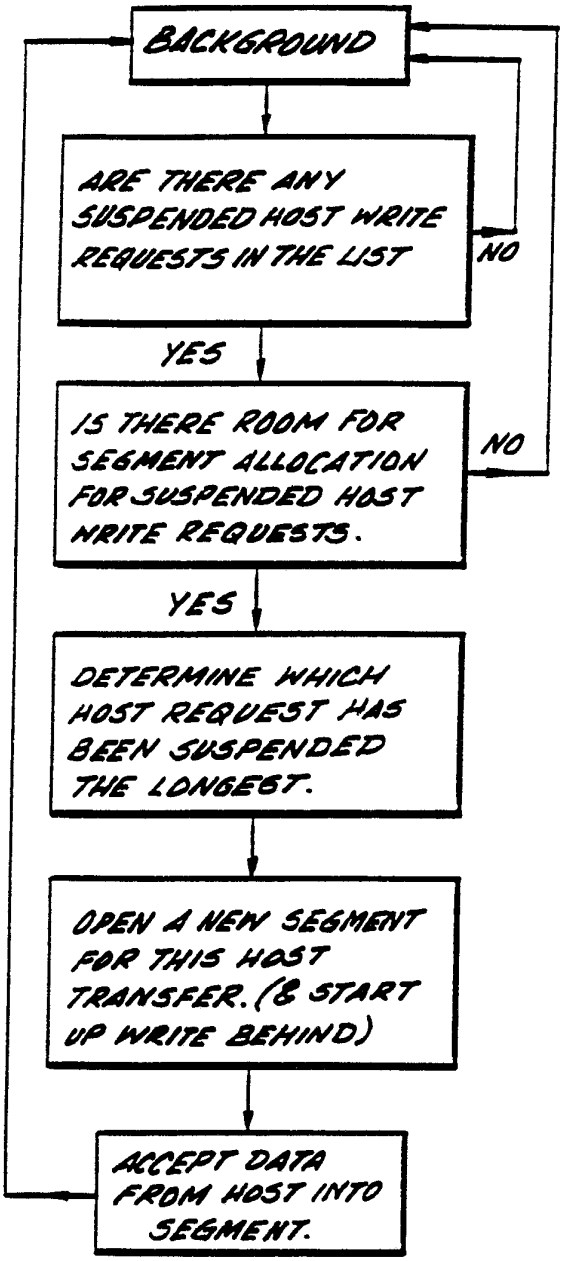

BACKGROUND
ARE THERE ANY SUSPENDED HOST WRITE REQUESTS IN THE LIST
NO
YES
IS THERE ROOM FOR SEGMENT ALLOCATION FOR SUSPENDED HOST WRITE REQUESTS.
NO
YES
DETERMINE WHICH HOST REQUEST HAS BEEN SUSPENDED THE LONGEST.
OPEN A NEW SEGMENT FOR THIS HOST TRANSFER. (& START UP WRITE BEHIND)
ACCEPT DATA FROM HOST INTO SEGMENT.

FIG. 11.

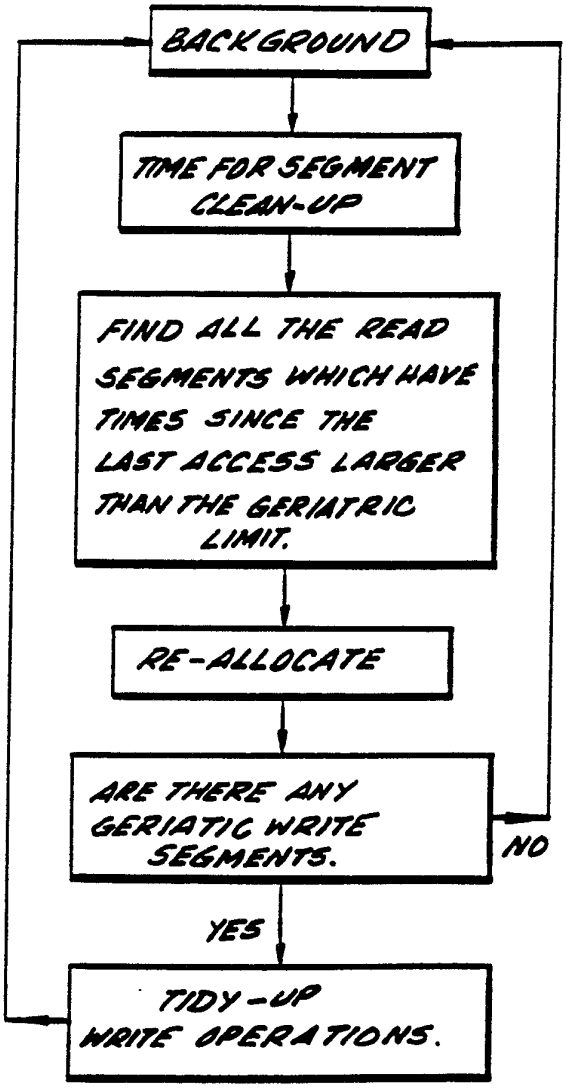

BACKGROUND
TIME FOR SEGMENT CLEAN-UP
FIND ALL THE READ SEGMENTS WHICH HAVE TIMES SINCE THE LAST ACCESS LARGER THAN THE GERIATRIC LIMIT.
RE-ALLOCATE
ARE THERE ANY GERIATIC WRITE SEGMENTS.
NO
YES
TIDY-UP WRITE OPERATIONS.

FIG. 12.



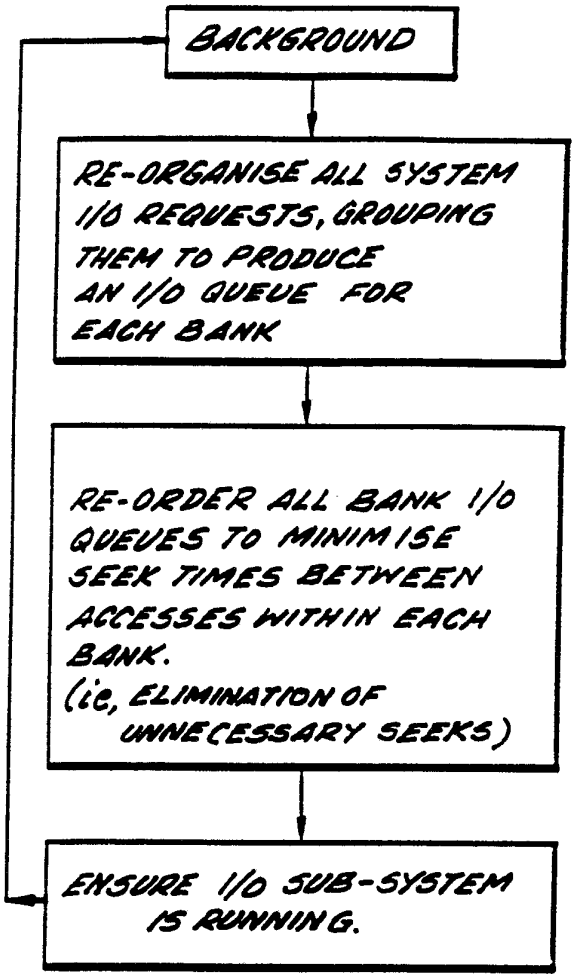
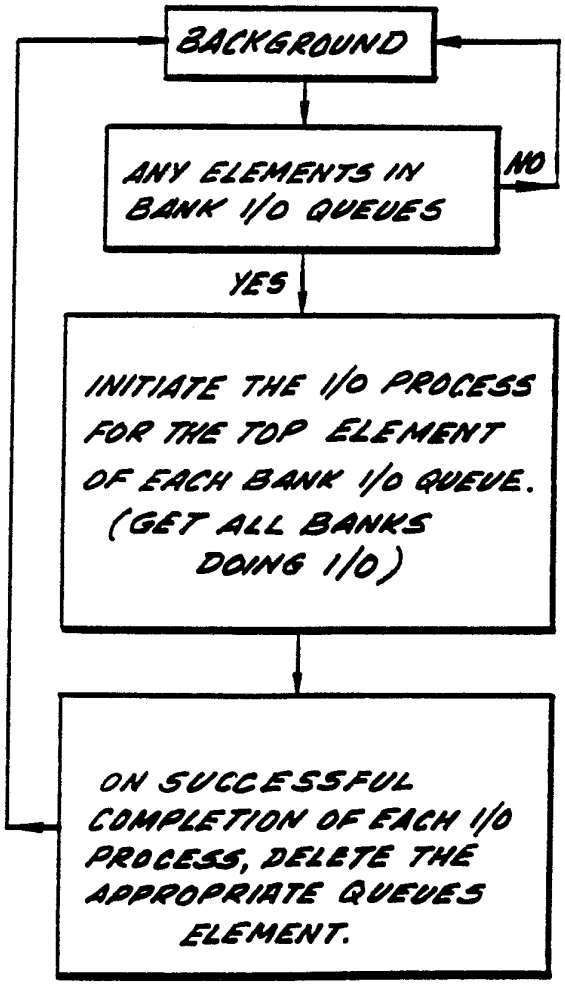

BACKGROUND
RE-ORGANISE ALL SYSTEM I/O REQUESTS, GROUPING THEM TO PRODUCE AN I/O QUEUE FOR EACH BANK
RE-ORDER ALL BANK I/O QUEUES TO MINIMISE SEEK TIMES BETWEEN ACCESSES WITHIN EACH BANK. (ie, ELIMINATION OF UNNECESSARY SEEKS)
ENSURE I/O SUB-SYSTEM IS RUNNING.

FIG. 13.


BACKGROUND
ANY ELEMENTS IN BANK I/O QUEUES
NO
YES
INITIATE THE I/O PROCESS FOR THE TOP ELEMENT OF EACH BANK I/O QUEUE. (GET ALL BANKS DOING I/O)
ON SUCCESSFUL COMPLETION OF EACH I/O PROCESS, DELETE THE APPROPRIATE QUEUES ELEMENT.

# INTERNATIONAL SEARCH REPORT

International Application No. PCT/GB 91/01557

## I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all)[6]

According to International Patent Classification (IPC) or to both National Classification and IPC

Int.Cl. 5 G06F12/08; G06F3/06; G06F11/10

## II. FIELDS SEARCHED

Minimum Documentation Searched[7]

| Classification System | Classification Symbols |
|---|---|
| Int.Cl. 5 | G06F ; G11B |

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched[8]

## III. DOCUMENTS CONSIDERED TO BE RELEVANT[9]

| Category ° | Citation of Document,[11] with indication, where appropriate, of the relevant passages[12] | Relevant to Claim No.[13] |
|---|---|---|
| Y | WO,A,8 909 468 (UNISYS CORPORATION) 5 October 1989<br><br>see page 9, line 33 - page 12, line 3<br>see page 15, line 1 - page 19, line 3; figures 5-8 | 1-3,6, 8-11, 13-15,21 |
| | --- | |
| X | EP,A,0 369 707 (ARRAY TECHNOLOGY CORP) 23 May 1990 | 18-19 |
| Y | see column 5, line 33 - column 6, line 37<br><br>see column 7, line 48 - column 8, line 47<br>see column 11, line 24 - line 49 | 1-3,6, 8-11, 13-15,21 |
| | --- | |
| | -/-- | |

° Special categories of cited documents :[10]

"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

"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

## IV CERTIFICATION

| Date of the Actual Completion of the International Search | Date of Mailing of this International Search Report |
|---|---|
| 13 JANUARY 1992 | 21. 01. 92 |
| **International Searching Authority** | **Signature of Authorized Officer** |
| EUROPEAN PATENT OFFICE | MOENS R.A. |

III. DOCUMENTS CONSIDERED TO BE RELEVANT (CONTINUED FROM THE SECOND SHEET)

| Category ° | Citation of Document, with indication, where appropriate, of the relevant passages | Relevant to Claim No. |
|---|---|---|
| X | WO,A,8 910 594 (AMDAHL CORP) 2 November 1989 | 18-19 |
| A | see page 6, line 18 - page 8, line 22<br>see page 19, line 23 - page 22, line 9<br>see page 27, line 21 - page 28, line 23<br>see page 29, line 11 - line 23; figures 2,5,6 | 1,9,13, 14,21 |
| A | EP,A,0 278 425 (IBM) 17 August 1988<br>see column 3, line 52 - column 6, line 37; figures 2-4 | 1,4,9, 12-13, 15,16 |

# ANNEX TO THE INTERNATIONAL SEARCH REPORT ON INTERNATIONAL PATENT APPLICATION NO. GB 9101557 SA 51311

This annex lists the patent family members relating to the patent documents cited in the above-mentioned international search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information. 13/01/92

| Patent document cited in search report | Publication date | Patent family member(s) | Publication date |
|---|---|---|---|
| WO-A-8909468 | 05-10-89 | US-A- 4958351 | 18-09-90 |
| | | EP-A- 0362362 | 11-04-90 |
| | | JP-T- 2503966 | 15-11-90 |
| EP-A-0369707 | 23-05-90 | AU-A- 4452989 | 17-05-90 |
| | | CA-A- 2002750 | 14-05-90 |
| | | JP-A- 2236714 | 19-09-90 |
| WO-A-8910594 | 02-11-89 | US-A- 4993030 | 12-02-91 |
| | | AU-B- 614611 | 05-09-91 |
| | | AU-A- 3448389 | 24-11-89 |
| | | AU-A- 7935991 | 03-10-91 |
| | | EP-A- 0414729 | 06-03-91 |
| EP-A-0278425 | 17-08-88 | JP-A- 63204448 | 24-08-88 |