An Overview of the Yale GEM System

(Preliminary Version)

P. Weiner, C. R. Minter

Research Report #18

This work was partially supported by grants from the Alfred P. Sloan Foundation and the Exxon Education Foundation.

April 1973

An Overview of the Yale GEM System

Abstract

The Yale GEM system -- a Graphical Editing Machine -- aims to provide low cost, highly interactive computing to undergraduates. This report discusses the philosophy behind the system, architectural issues, and a few specific details.

I. Introduction

Since 1970, the Department of Computer Science at Yale has been engaged in an effort to develop and demonstrate low cost, highly interactive computing techniques appropriate for the university setting. The first major undertaking by the department was the acquisition of an off-the-shelf computer (a PDP-10) and character display CRT terminals (Sugarman model T6). Using software supplied with the computer, augmented only by an editing program designed to take advantage of the two-dimensional capabilities of the CRTs, this system is managed by the department and provides both educational and research computing to a large segment of the Yale community. The cost of an hour's session on the machine is between $2.50 and $5.00.

The unquestioned success of the Yale PDP-10 has led the department into a more ambitious project: to design and deploy a computer system that provides educational computing to undergraduates at an hourly cost of between 25 and 50 cents per hour. Moreover, the department aims to parallel this cost reduction with an increase in interactive capability. In particular, limited graphics facilities are provided, allowing, for example, variable type fonts as well as the display of pictures.

The GEM system under development at Yale aims to achieve this reduction in cost, together with the increase in interactive capability. The system -- a Graphical Editing Machine -- has been designed to be

what its name suggests.

Almost all of the design specifications of the GEM system can actually be realized with off-the-shelf hardware, but not at the cost level we have set. Indeed, we have found it necessary to undertake extensive hardware (and software) work. Standard off-the-shelf components are used in the GEM system, however, whenever cost considerations permit. New technologies are also employed, but only when strong evidence exists that they will be commercially available as standard components in the near future.

In the design of any system, it is necessary to trade off cost considerations for capabilities. We follow a simple rule when such decisions arise: Calculate the cost of providing the essential capabilities and do not add any improvement, no matter how significant, if the cost rises more than a nominal amount (say ten percent). While the proposed capabilities of the GEM system exceed that of most modern student time-sharing systems, it is not hard to imagine enhancements that, while not without value, add significantly to the cost. We have been firm in rejecting these enhancements.

Indeed, we designed the GEM system by first selecting the essential components of a terminal system which permit limited graphical capability at minimum cost. It is important to observe that a GEM computer system is but a natural extension of a GEM terminal system. That is, the basic components of the GEM system can be used to provide

inexpensive graphics without comprehensive computing, and that the computing capabilities of the GEM system come at relatively little incremental cost.

The next section describes the essentials of a GEM terminal system, and is followed by a discussion of ways to augment these basic components to obtain a comprehensive computer system. The remainder of the paper addresses architectural issues; the emphasis is on the hardware aspects of the system. Software considerations played a significant role throughout the hardware design, but we describe software details elsewhere.

II. Essentials of the GEM terminal system

Almost by definition, a GEM system must contain a two dimensional output device capable of displaying a sizable amount of text as well as graphical information with reasonable resolution. A keyboard is required to communicate with the system, and either special purpose or general purpose hardware must be present to carry out the functions of terminal control.

There are two inherently different methods for displaying material on a two dimensional surface. The first uses vector line segments, and the second forms images as a collection of dots, closely spaced together. Cost considerations rule out vector systems, with the possible exception of storage-tube devices. But while these displays are cheap enough, we do not feel they have the interactive capability or speed required for flexible text editing. Several new technologies support dot representation display, for example, plasma panels. It may well be that some one of these technologies will eventually dominate the information display field; for the moment, it seems better to watch with interest from a distance.

Ordinary raster-scan TV technology also supports dot representation display. Indeed, 20 lines of eighty characters may be displayed using 240 horizontal bit lines and 640 vertical bit lines. If this is done, then 9600 16-bit words must be stored in one form or

another. Different numbers of bit lines are also possible, and indeed may be preferred if the emphasis is on graphics applications.

One approach to storing the bit images is to use rotating magnetic storage mediums or solid-state shift register circuits. Both methods require a special purpose processor to keep track of the information on the medium and provide flexible methods of access and modification. Another approach is to use random access memory. This method enjoys the advantage of convenient access and modification, and permits the use of an off-the-shelf minicomputer for purposes of terminal control. Indeed, use of random access memory allows other parts of the system to be simplified considerably. Happily, the use of this form of memory does not entail a price disadvantage; the plain fact is that with the emergence of random access MOS integrated circuits, the current price, under a penny per bit, is no higher than that of less flexible technologies.

Thus a basic GEM system consists of TV sets, standard keyboards, a small minicomputer (for example, a PDP-11/05), and a large amount of memory. The prototype GEM supports 16 terminals, and has a 16K memory segment for each terminal. Our somewhat novel use of this memory tempts us to occasionally refer to it as the system's GEMory.

III. Peripheral devices in the GEM system

To serve as a comprehensive computer system, several peripheral devices must be added to the basic GEM system. First, some form of hard copy must be available. Second, provision must be made to allow a student to save and restore information created on the system. Finally, a file system must be provided to store the files used during a session on the computer. For each requirement, several alternative technologies are available. We discuss some of the choices, together with our current evaluations.

A standard line printer can be used to provide minimal hard copy output for the student. A much better idea is to employ a dot-matrix printer. With a device of this kind, anything displayable on the TV screen can be replicated on paper. The cost of these devices seems to be dropping rapidly, and may soon be less expensive than more traditional devices. The only major disadvantage of dot-matrix printers is that special paper is used, and current prices are high, but this is expected to change as production volume is increased. Indeed, we anticipate that such a device will be a standard component of a GEM system.

Our experience on the PDP-10 has confirmed that mini-tapes are very useful in allowing the student to save and restore files that are created on the system. Similar, but less expensive, tape cassettes

are now being marketed. Reliability is an absolutely critical requirement, and only time will show if these devices measure up. Another device, with unknown reliability, is the floppy disk. Comparable amounts of information are stored on a flexible record-size disk, and may be easily mounted into a disk drive. The performance specifications of these disks far exceed those of tape cassettes. If reliability can be demonstrated, the floppy disk will dominate the other approaches to file entry. Preliminary indications suggest good reliability. We plan to have at least two in our prototype.

Another interesting use of floppy disks is for secondary storage. Here, each student is provided with his own drive, and once his disk is mounted he works on the system having his own on-line file structure. The performance of these devices in an editing environment is aided by the fact that each has an independently controllable head which may be pre-positioned to any track. But new mass storage devices announced quite recently, with very large storage capabilities on the order of 40 mega-bytes, may tilt the secondary storage decision in favor of one high performance disk. However, a small GEM system with but a few terminals may still find it advantageous to use the floppy disks for both file entry and file storage.

Every GEM system should have at least one large computer available as a "peripheral device." In our own case, we will connect our GEM to the PDP-10 through a high-performance communication link. Our PDP-10

is itself connected to a remote CDC 6600 and other large machines.

IV. Architecture of the GEM computer system

Perhaps the first important design decision is to choose the resolution of display and the associated size of the GEMory. Of course, if only character display is required, only a very small amount of GEMory is required (assuming that hardware character generators are built into the system). But we consider graphical capabilities to be essential.

Text editing and other computations require workspace memory. Previous experience indicates that 20 lines of text is an adequate, though minimal, amount of text to display during editing. These considerations suggest a resolution of 240 bit lines by 640 bit lines, which is compatable with standard TV (non-interleaved) conventions. Bit density along a horizontal line is exactly twice that along a vertical line if the normal three to four aspect ratio is employed. Note also that the use of 640 vertical bit lines allows eighty characters to be displayed with each character in an eight bit field. This means that software character generation is possible without the need for shifting or masking by a computer with 8-bit bytes.

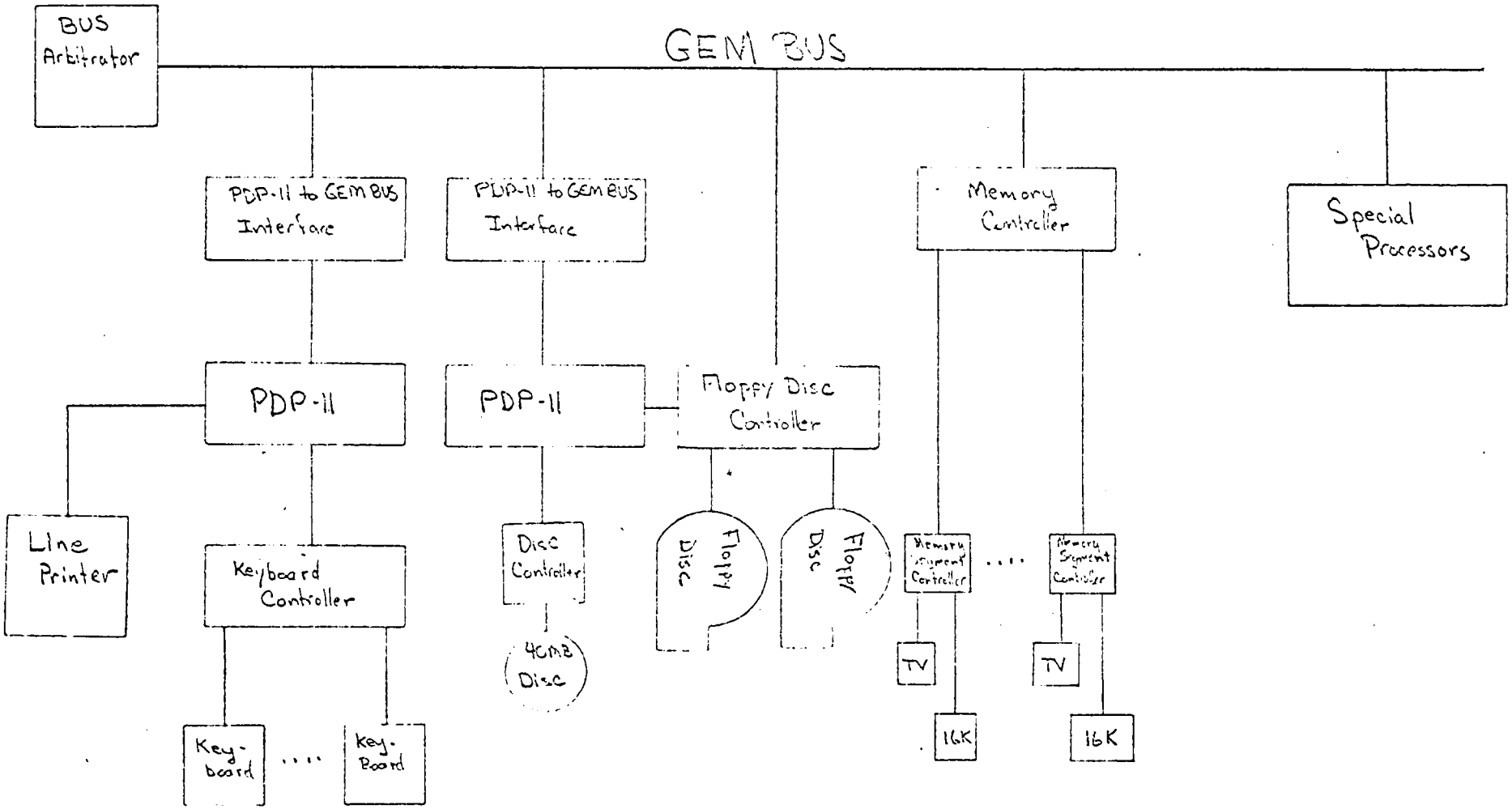
The total amount of GEMory required for terminal display is 9600 16-bit words. This means that the selection of a 16K GEMory segment leaves over 6K of memory for workspace. Software considerations support the choice of this amount, which also permits convenient GEMory organization.

It is worth emphasizing that the GEMory size decision is dictated by essential requirements of the GEM system. Much higher resolution is technically possible, and in some circumstances quite useful, but only at a significant increase in cost. This cost cannot be justified for the student application we address.

The second major design decision regards that of computational capability. The desire to have easy expansion of computer power suggests that several processors be used rather than one large processor. Since each processor needs convenient access to the central GEMory, a bus structure is employed. The refresh of the TV screens does not pass through this bus, and indeed our organization ensures that the GEMory is available to the computers and other devices on the bus most of the time. Both bus and memory speeds are quite fast, so several computers and devices can coexist without performance degredation.

At any given time, only one segment of GEMory can be directly addressed by each computer on the bus. Context switching between segments is conveniently done by loading a single register. We call the bus the GEMbus. Figure 1 shows the over-all organization of the GEM system.

11.



An overview of the GEM SYSTEM.

Figure 1.

V. GEMbus conventions

For two reasons, the GEMbus closely resembles the PDP-11 UNIBUS. First, several PDP-11s are connected to the GEMbus via an interface between the GEMbus and the computers' UNIBUS. The complexity of this interface is minimized by adopting similar conventions. Second, the similarity of the two bus conventions makes it possible to convert a UNIBUS interface to a GEMbus interface very easily. Specifically, this means that interfaces to the GEMbus may be built and checked out on a UNIBUS before construction of the GEMbus is complete.

An important difference between the GEMbus and a UNIBUS is the number of address lines. While only 18 lines are provided on a UNIBUS, we include 24 address lines on the GEMbus. In addition, there is a separate bus arbitration unit that allows connection of several independent computers. This unit allows control of the GEMbus to shift from one processor to another without need for direct communication between the two processors.

There are two types of operations that take place on the GEMbus -- data transfers and control transfers. The bus is set up so that these two types of operations can occur simultaneously; a new bus master may be selected during a data transfer operation. There are four signals involved in the transfer of bus control from one device to another:

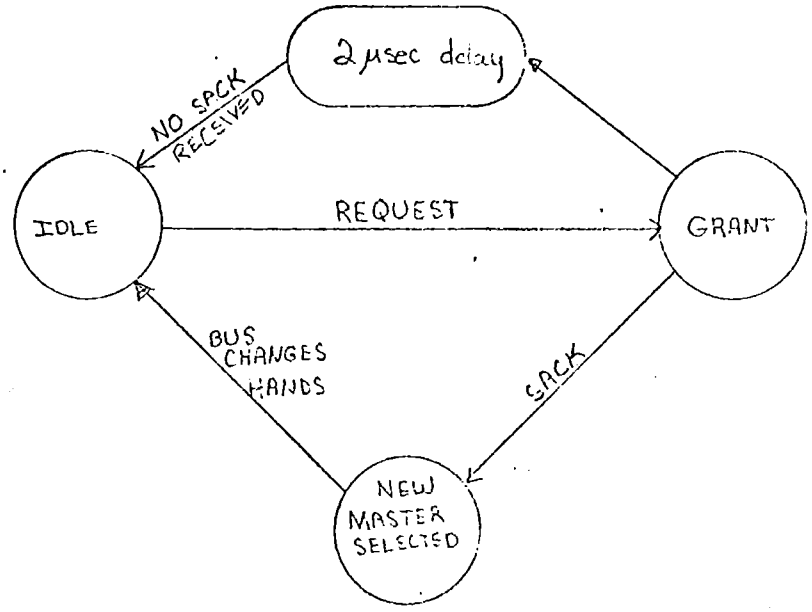
1. the request line

2. the grant line
3. the grant acknowledge line
4. the busy line.

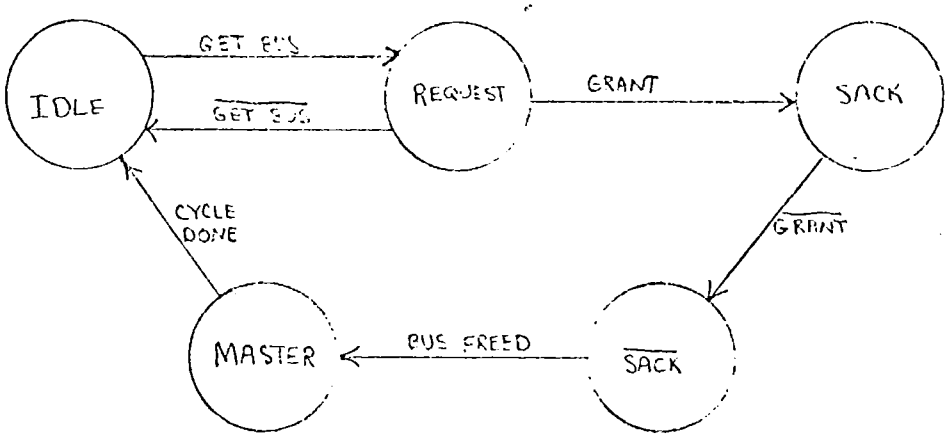
When a device requests the use of the bus and the next owner has not been selected, the arbitration unit sends out a grant to the requesting device. If the next owner has already been selected the arbitration unit waits until the previously selected user takes control of the bus before issuing the grant. The device selected takes control of the bus after the current bus master releases it. Figure 2 shows a simplified version of the bus arbitrator.

Data transfer operations on the GEMbus are quite similar to those on a UNIBUS. When a device becomes master, it asserts address and control information (read or write), and the data (in the event of a write operation). After a delay to insure that all signals have settled, the master asserts a signal that tells all potential bus slaves that the bus lines are valid. At this point, the referenced device does one of two things. If the current operation is a write, the selected device takes the information from the bus, places it into a buffer register, and then notifies the bus master that the data operation is complete. (The information in the buffer register may be transferred to memory after the bus cycle is finished.) If the operation is a read, the selected device gets the data requested by the master, places it on the data lines of the bus, and notifies the bus master that the data is valid.

Transfer of BUS ownership



BUS Arbitrator



Requesting Device

Figure 2

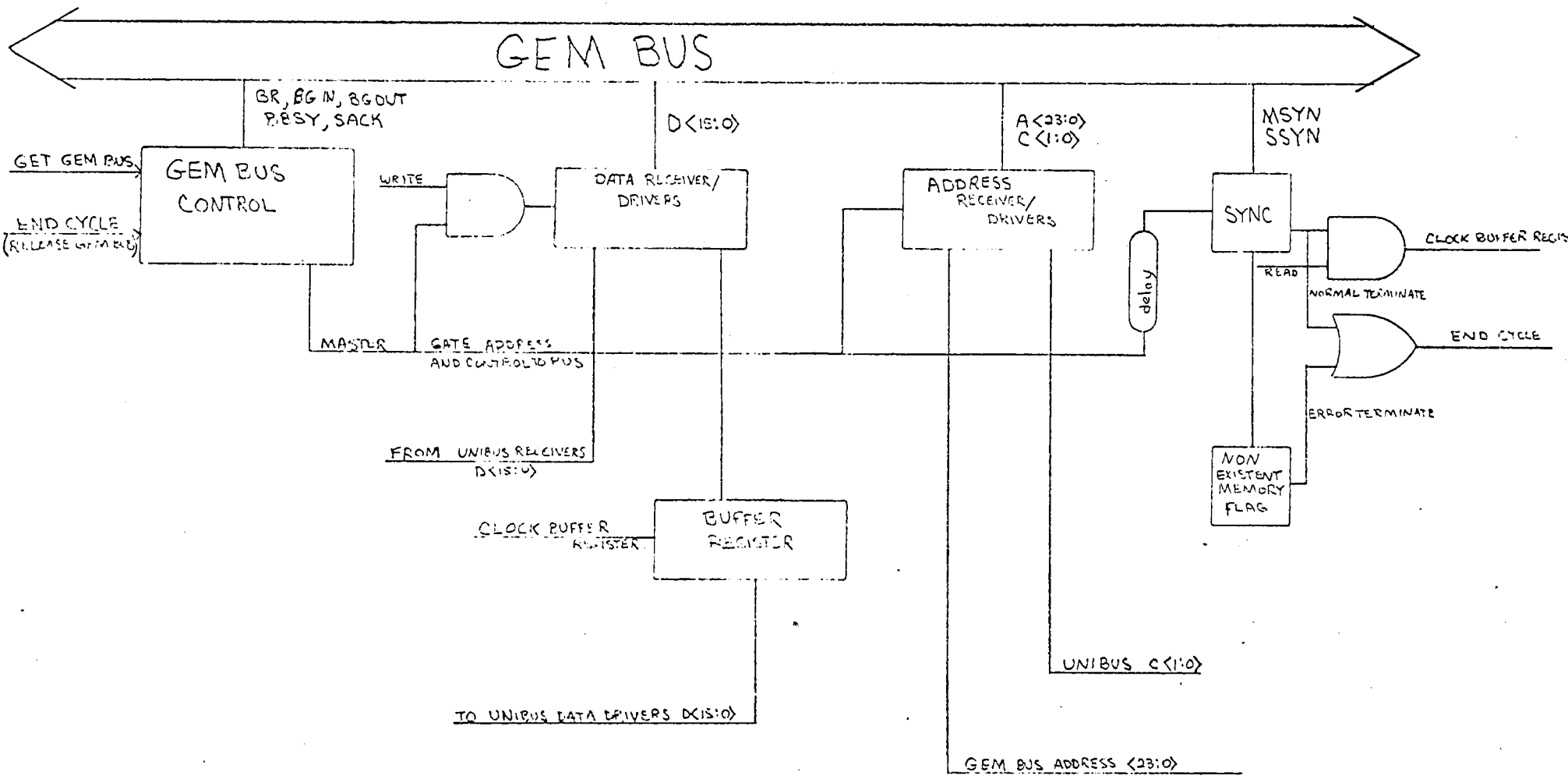
The bus master picks up the data, terminates the cycle, and releases the bus.

VI. The UNIBUS to GEMbus interface

The UNIBUS to GEMbus interface contains circuitry to allow the UNIBUS to become master of the GEMbus as well as circuitry to conduct a data transfer operation. These functions are illustrated in Figure 3.

As shown in the figure, all transfers must be initiated by the PDP-11; devices on the GEMbus can reference the PDP-11's memory only if a separate interface is added. The entire cycle is started by the signal called GET GEMbus which goes into the box marked GEMbus CONTROL. This box then goes through the sequence of requesting the bus, receiving the grant, acknowledging the grant, and taking control of the bus the next time it becomes free. When the GEMbus CONTROL module informs the other modules that it is now bus master, several things happen. First, the address and control lines (generated elsewhere in the interface) are gated onto the GEMbus. If the operation is a write, the data from the UNIBUS is also gated onto the GEMbus. After a short delay to ensure the validity of the data, the interface asserts MASTER SYNC on the GEMbus. When it receives a SLAVE SYNC, it terminates the bus cycle and releases the GEMbus. If no response is forthcoming, the interface assumes that no such address exists on the bus, and the cycle is terminated with the NXM (non-existent memory) flag set.

Other parts of the UNIBUS to GEMbus interface perform the address translation from the 18-bit address space of the UNIBUS to the 24-bit

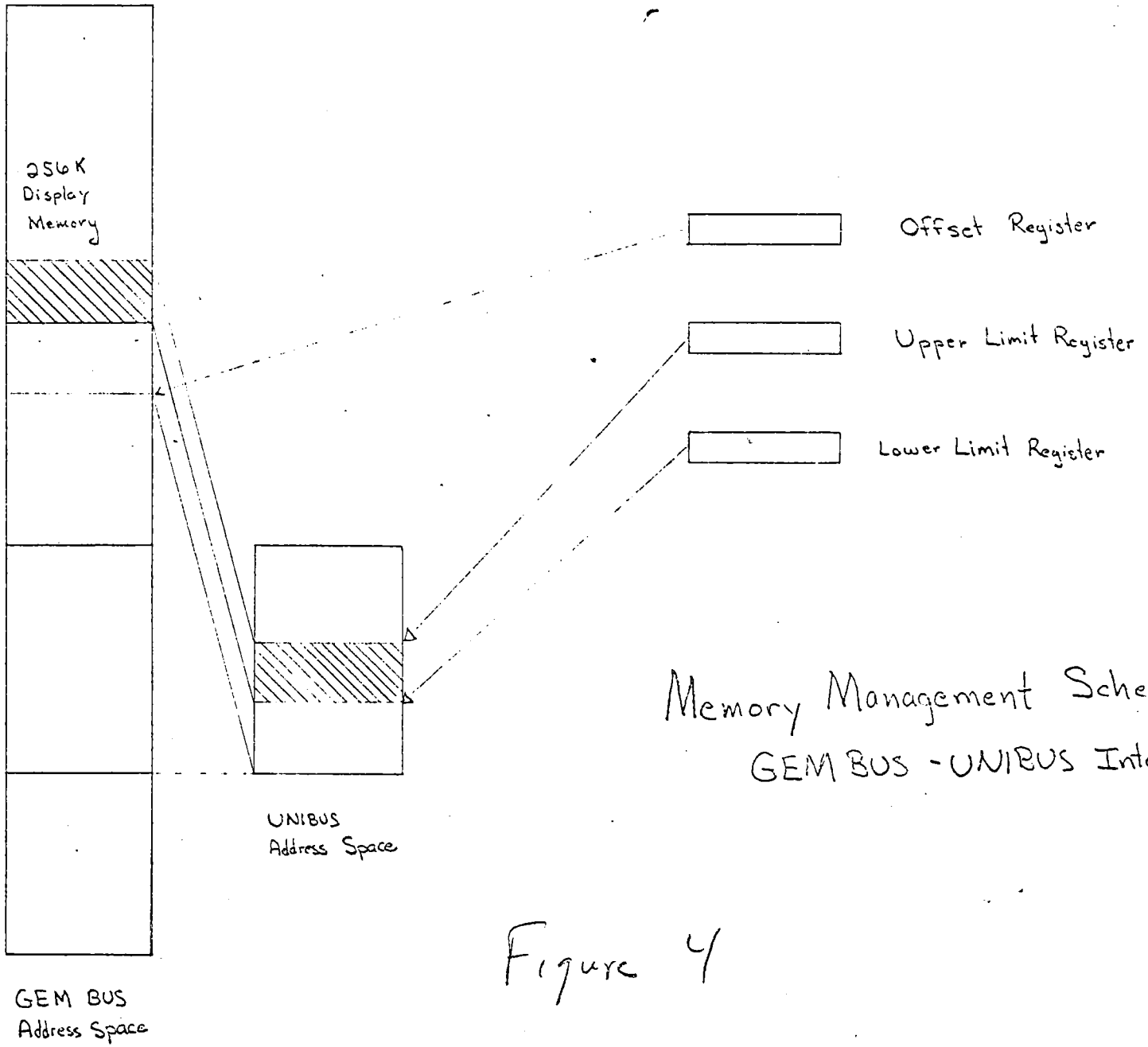


UNIBUS- GEMBUS INTERFACE

GEMBUS CONTROL

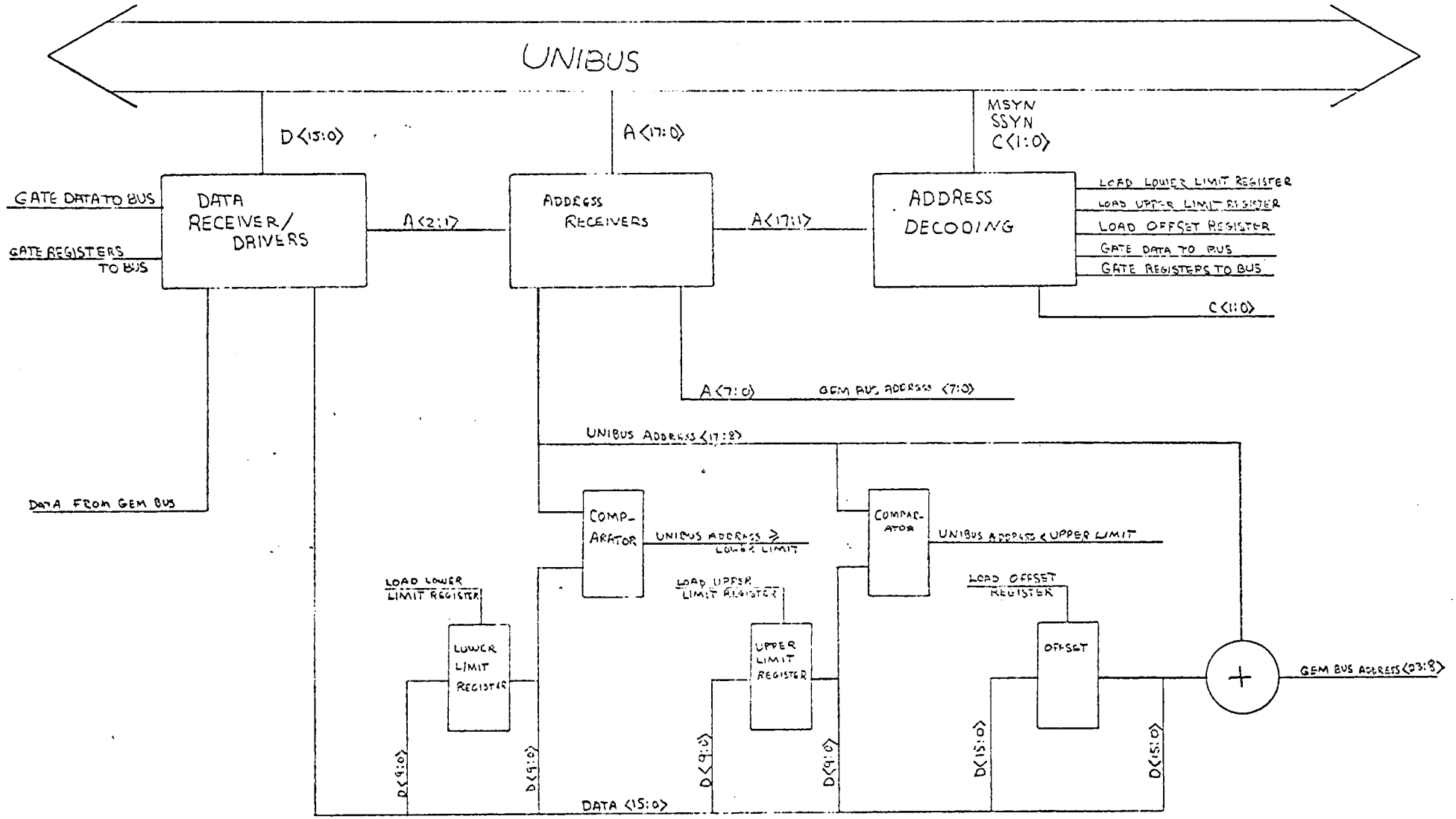
Figure 3

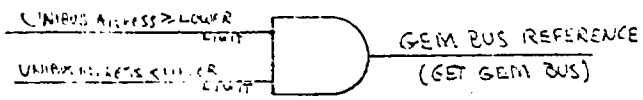
address space of the GEMbus, as shown in Figures 4 and 5. Note that three registers are used. The first two, the lower limit register and the upper limit register, are each ten bits long. The contents of these two registers are constantly compared with the ten high order address bits on the UNIBUS. Whenever the address on the UNIBUS is between these two limits, the interface treats the current UNIBUS cycle as a reference to the GEMbus. The GEMbus address is computed by adding the contents of the 16-bit offset register to the ten high order bits of the 18-bit UNIBUS address, yielding the high order 16 bits of the GEMbus address. The low order eight bits of the UNIBUS address are passed directly to the GEMbus, as are the two control bits, C0 and C1. Note that this method of mapping allows a PDP-11 to fill any part of its 28K address space not taken up by its own memory with memory on the GEMbus; the filled-in part consists of an integral number of 128-word blocks relocated to an even 128-word boundary.



Memory Management Scheme
GEM BUS - UNIBUS Interface

Figure 4





UNIBUS - GEMBUS INTERFACE
ADDRESS MAPPING

Figure 5

VII. The GEMbus to UNIBUS interface

The UNIBUS to GEMbus interface allows the UNIBUS to become master of the GEMbus, but not the other way round. Actually, if the interface is made symmetrical with respect to control, it is possible for the entire system to hang up in a "deadly embrace," unless special provisions are made.

To see why this is so, suppose there are two PDP-11s on the GEMbus, and that each wishes to read data from the memory of the other. Each initiates UNIBUS cycles, but only one can become master of the GEMbus, say the first. Now the GEMbus cannot do an NPR (non-processor) reference of the second PDP-11's memory because its UNIBUS is busy, and it will stay busy so long as the GEMbus is busy. Clearly, the three buses are locked up. Hardware timers will break the deadlock, but at the expense of an error trap. Clearly, the situation should not be treated as an error condition.

One way to prevent such traps is to insert a set of switches in each UNIBUS, placing devices which can be bus masters on one side and NPR targets on the other. These switches are opened whenever a UNIBUS requests control of the GEMbus, and the GEMbus is referencing the UNIBUS. Note that all cycles eventually complete and no timeout traps occur.

In addition to the special circuitry described, the GEMbus to

UNIBUS interface contains rather standard DMA components. These allow NPR transfers to take place whenever the address on the GEMbus is in an address window assigned to the PDP-11.

VIII. The GEMory controller

Our 16-terminal GEM prototype has 256K of GEMory organized in 16 16K segments, one per terminal. The memory is constructed from 1024-bit MOS RAM integrated circuits whose cycle time is 450 nanoseconds and whose access time is 325 nanoseconds. Part of the memory interface is replicated 16 times, and part is common. The interface connects the memory to the GEMbus, but it also controls the refreshing of the TV screens. The data rates for refreshing are rather high -- each TV screen needs a new data bit every 80 nanoseconds. But since each TV set may be fed in parallel, this requires only one memory fetch every 1.3 microseconds. Even so, the memory would be constrained from responding to requests from other high speed devices on the GEMbus, if direct memory fetches were permitted. Instead, we have organized the memory to permit four fetches in parallel (for each terminal!), and the memory is consequently hit by the refresh circuitry only once every five microseconds. Moreover, the memory demands of the TV sets are predictable, and we are able to guarantee that the memory is available for refresh purposes without disturbing the normal asynchronous mode of the GEMbus.

The shared components of the interface are relatively simple. They detect when the GEMbus wishes to reference the memory, and decode the address to determine which of the 16 segments is being referenced.

The address is passed to the selected segment, together with any data (in the event of a write). When the segment acknowledges selection, the GEMbus cycle is completed.

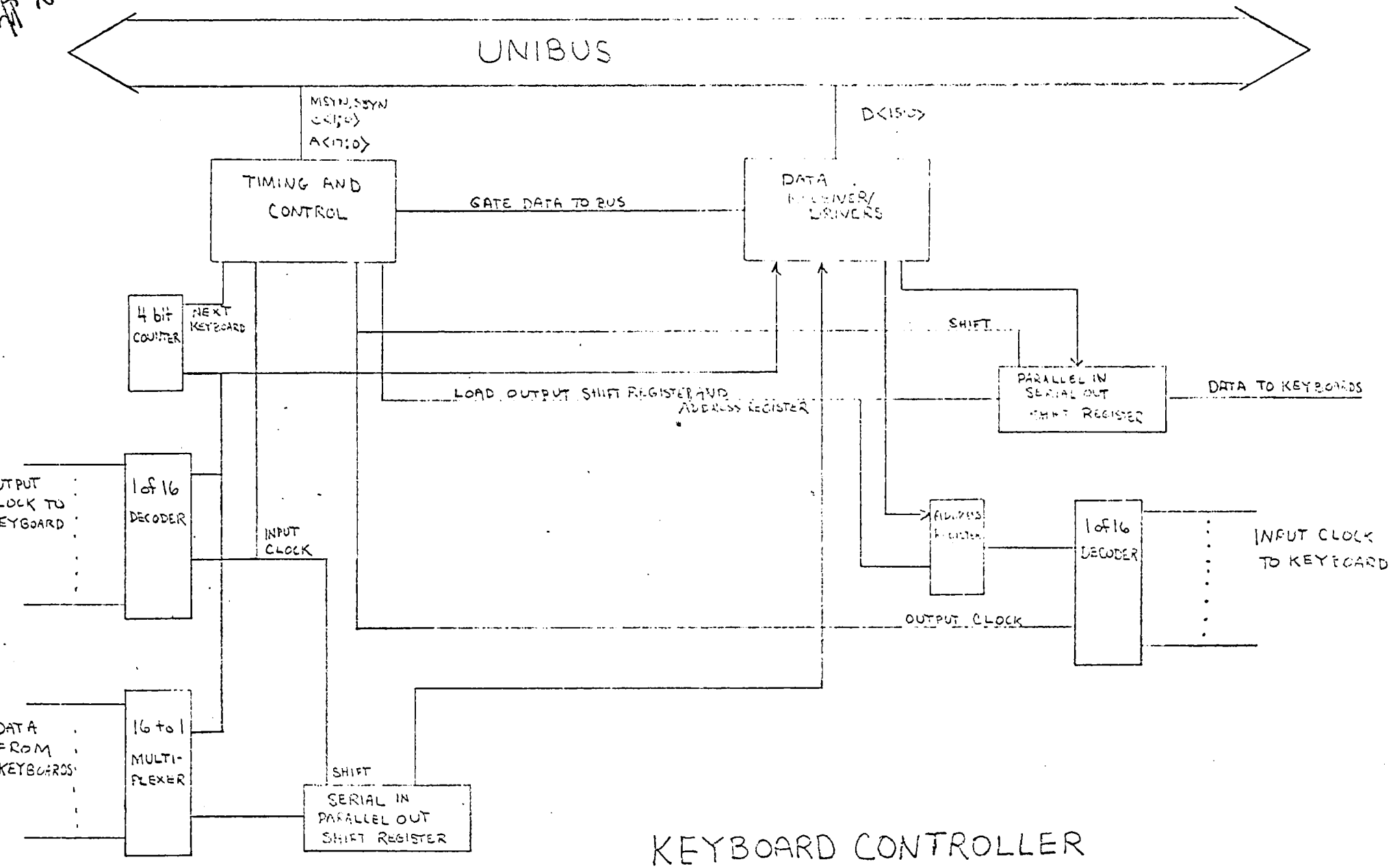
There is also common circuitry to provide timing and control of the TV sets. A timing chain is started when the TVs require more data. After about 4.3 microseconds, new memory cycles are inhibited. Another 450-500 nanoseconds are skipped to allow current memory cycles to terminate. At this point, four words are fetched from each 16K segment simultaneously and loaded into a 64-bit shift register.

The replicated components contain the display shift registers and video drivers as well as the circuitry necessary to handle the memory itself. To conserve I/O connections, the 64-bit shift register is loaded in 16-bit blocks at 80 nanosecond intervals.

IX. The keyboard controller

The GEM system uses standard keyboards as input devices. The sixteen keyboards are connected to a PDP-11/05 through a common controller, as illustrated in Figure 6. The controller is completely separate from the display controller -- any association of a keyboard to a TV screen must be made by software. Characters typed on the keyboards are transmitted to the computer. In addition, data may be passed from the computer back to the keyboard to control four pairs of status lights and a buzzer.

The keyboards contain a parallel-in serial-out shift register that is used to serialize the 11-bit codes generated by a keystroke. (Each button generates 6 bits and in addition 5 shift keys may be pressed.) A bit is shifted out of the register upon receipt of a strobe command from the controller. The presence of a character is indicated to the controller by the value of the low-order (first out) bit in the register. This bit is normally set to 1, but when a key is struck it is reset to 0. The common controller sequentially scans each register, testing the low order bit, and determines if a character is present. When a zero value is seen, timing and control circuitry sends a succession of strobe signals to pass the contents of the keyboard register to a central register. When the entire code has been passed, the terminal number is appended and an interrupt of the computer is



KEYBOARD CONTROLLER

Figure 6

requested. Finally, the controller continues scanning the keyboards beginning with the next one in sequence.

The computer to keyboards part of the controller consists of a parallel-in serial-out shift register, a one-of-sixteen decoder, an address register, and timing and control circuitry. The computer loads the shift register and indicates the keyboard number by loading an address register at the same time. Timing and control circuits send the bits serially to an internal register in the keyboard. This register controls the display lights and the buzzer.

X. The floppy disk controller

This controller is a sophisticated micro-programmed processor capable of handling up to 16 floppy disks. By using a micro-processor it becomes possible for the floppy disks to share the same DMA logic and to include several other very useful features.

The detailed operation of the floppy disk controller is too involved to be covered here; we present only a brief description of its most important capabilities. The controller is an interface between the GEMbus and up to 16 floppy disks, controlling the transfer of data between the disks and any slave device on the bus as specified by a command list. The four main commands are to read a sector into memory, read a sector and compare with memory, write a sector from memory, and seek a track. The command list of each disk is stored in the GEMory and is automatically fetched by the disk controller.

Upon completion of a command the controller will either fetch another command for the same disk, branch to another point in a command list, or inform the computer of the completion of a command list. Control information is sequentially scanned by the controller to permit initiation of a new command list, and 16 seeks can be done in parallel. However, only 4 disks can read or write simultaneously. This number is dictated by GEMbus efficiency considerations. Note that the computer has the capability of monitoring the progress of the controller and

the command lists may be dynamically altered. This is an important advantage since the computer can also read the rotational position of each disk, and optimization based on this information is often significant.

A more detailed description of the floppy disk controller will be forthcoming in another paper.

XI. Personnel and acknowledgements

Many people in the Department of Computer Science at Yale have contributed significantly to the development of the GEM system, and the attempt here to give credit cannot possible be considered exhaustive. The project is part of a larger effort supported by the Alfred P. Sloan Foundation and the Exxon Education Foundation. Peter Weiner has served as principal investigator of this effort. Charles Minter coordinated the hardware work. He also designed and constructed most of the interfaces. Inder Singh, who first suggested the use of random access memory, designed the keyboard interface. Jack McDonald is responsible for the floppy disk work, Robert Tuttle for a PDP-11 to PDP-11 interface, and John Lewis for an interface to a Versatec dot-matrix printer. Jim Sustman, John Wick, Fred Howard, and Ron Harris also contributed to our hardware effort.

Mention should be made of the helpful suggestions of John McCarthy of Stanford and Tom Knight of MIT. Finally, Ned Irons and Alan J. Perlis of Yale must be thanked for their encouragement and support.