A TIME DIVISION MULTIPLEXER/DEMULTIPLEXER FOR AN EXPERIMENTAL MULTIPLE ACCESS LIGHTWAVE SYSTEM

by

LOUIS ANDREW NAGODE

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREES OF
BACHELOR OF SCIENCE
and
MASTER OF SCIENCE
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June, 1981

(c) Louis Andrew Nagode, 1981

The author hereby grants to MIT and Bell Telephone Laboratories, Incorporated permission to reproduce and to distribute copies of this thesis document in whole or in part.

Signature of Author ______________________
Department of Electrical Engineering and Computer Science, May 8, 1981

Certified by ______________________
Robert S. Kennedy, Thesis Advisor

Certified by ______________________
A. Albanese, Company Project Supervisor (VI-A Cooperating Company)

Accepted by ______________________
Chairman, Departmental Committee on Graduate Students

Archives
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
MAY 27 1983

A TIME DIVISION MULTIPLEXER/DEMULTIPLEXER FOR AN EXPERIMENTAL MULTIPLE ACCESS LIGHTWAVE SYSTEM

Louis Andrew Nagode

ABSTRACT

A time division multiplexer/demultiplexer for use in an experimental multiple access lightwave network is described. The multiplexer circuit capable of handling data rates up to 100 Mb/sec was built using state of the art emitter coupled logic. This multiplexer is able to absorb a continuous digital data stream at rates between 768 Kb/sec and 100 Mb/sec while at the same time retransmitting the data in 100 Mb/sec "bursts" during assigned 960 ns time slots. This design has direct applications for use in any multiple access networks using time division multiplexing. A description of the design and related components is included.

ACKNOWLEDGMENT

I would like to express my sincere appreciation to Andres Albanese for giving me many of the basic ideas for this project. He stimulated me to produce new ideas and he was always there to shed new light on areas I had previously not seen.

I would like to thank W. M. Hubbard, A. R. McCormick, H. E. Kehlenbeck, and C. A. Caroli for allowing me to draw from their previous experiences on many practical matters.

I would also like to express my appreciation to my Department Head Detlef Gloge for his concern and interest in my project.

The people at Bell Laboratories have truly made this assignment, as well as the previous ones, a wonderful experience.

Finally I wish to thank my fiancee Jennifer Phillips for all her loving support and aid in the preparation of this thesis.

Bell Laboratories

600 Mountain Avenue
Murray Hill, N J 07974
Phone (201) 582-5444

S. E. Hollander
General Legal and Patent Counsel

May 19, 1981

Department of Electrical Engineering
 and Computer Science
Room 38-444
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Attention: Professor A. C. Smith

Gentlemen:

Master's Thesis of Louis A. Nagode

The accompanying thesis by Louis A. Nagode has been reviewed and has been found to contain proprietary information that is currently under patent study at the Bell Telephone Laboratories, Incorporated. However, in order not to prevent Mr. Nagode from graduating, the Bell Telephone Laboratories, Incorporated has no objection to the limited release of the material to the extent necessary for this purpose. This limited release is granted with the understanding that there is agreement that the thesis will not be placed in the MIT library or otherwise made available to the public until the necessary patent study has been completed and a general release is granted, in writing, by the Bell Telephone Laboratories, Incorporated.

Thank you for your cooperation in this matter.

Very truly yours,

Attached
As above

TABLE OF CONTENTS

SECTION ONE: INTRODUCTION

As modern organizations grow into the world of data processing, the need for faster and more efficient data communication services will also grow. Due to inherent bandwidth limitations of conventional wire-pair lines, use of optical transmission media has been suggested for future data networks.[1] Data rates of 100 million bits per second (Mb/s) are readily achievable using available hardware. As the market-driven price of digital hardware declines, a digital optical fiber network will become increasingly more cost effective.

The following is part of a communication experiment designed at Bell Laboratories to transmit high-bit-rate signals over a local-area network for such services as facsimile, video, and high quality audio. Baud rates between 768 Kb/s and 98 Mb/s for virtual channels between two points on the network could be specified for the user, allowing him to tailor the network to his particular data communication needs. This thesis concentrates on the method of gathering data from the user and formatting this data such that it is possible to transmit the information over a distributed network in a circuit-switched mode. The following pages outline the format of this experiment and explain the digital hardware necessary to multiplex and demultiplex the data from a 100 Mb/s optical fiber channel.

SECTION TWO: A FIFO TUTORIAL

In the most general sense, both the multiplexer and the demultiplexer function as first-in, first-out (FIFO) buffers between the user and the network. In order to effectively introduce this topic, a brief tutorial will be provided on the structure of FIFO buffers.

The purpose of such a buffer is to absorb a particular sequence of information, then at some later time, releasing this information in the same order in which it was received. This sort of operation is generally achieved by using an addressable memory, an input register (to keep track of where the freshest piece of data is), and an output register (to keep track of where the oldest piece of data is). It is generally simplest to set up the device such that new data fills the memory sequentially, requiring only that the registers be incremented for each READ or WRITE operation. Due to memory limitations, however, it is necessary to reuse the memory locations, thereby changing the linear model of the FIFO device into a circular one. Thus when either register is incremented beyond the last location in memory, it will automatically jump to the first location. As data is pushed through the device, the output "pointer" consistently follows the input "pointer", and the two are always separated by the amount of data currently stored in memory (see Figure 1).

In section four, the implications and consequences of these limitation are reviewed in terms of what must be done to actually implement a FIFO model.

SECTION THREE: CHANNEL FORMAT

The data was formatted into 125 μs blocks in order to achieve compatibility with the experimental work currently in progress at Bell Laboratories Crawford Hill Laboratory. These 125 μs "frames" of data are further subdivided into 128 words, each of which is 96 bits long (see Figure 2). Multiplexers and demultiplexers are assigned 2^n word slots per frame ($n=0,\ldots,7$) depending on the bandwidth requirements of the user. In this way, the user has a menu of bandwidths (8 kHz X 96 bits/word X 2^n words = 768 X 2^n bits/s) from which he can choose an appropriate value for n. As the user's bandwidth requirement increases, the memory capacity of the multiplexer would necessarily have to increase in order to insure that the multiplexer/demultiplexer would not overflow with data that it is waiting to retransmit (either onto the optical fiber line or to the user). This is generally true <u>unless</u> careful decisions are made about word slot assignments.

A general scheduling algorithm for word assignments which would limit the maximum required memory size to be 2 words (192 bits) would be as follows. Supposing that the user has need for 2^{n_o} words/frame, the frame is then uniformly divided into $2^{(7-n_o)}$ "groups" of words and one word from each "group" would be assigned to each multiplexer/demultiplexer pair (see Figure 3). In this way, the multiplexer is assured that it will be able to retransmit a complete word of data (96 bits) by the time it has to completely absorb the next word of data. Keeping the memory size of the multiplexer/demultiplexer low is important because large memories for a distributed network would be both costly and too slow. An interesting scheduling algorithm that minimizes "blocking" in the network is presented in Appendix A.

This type of an overall format allows multiplexers and demultiplexers to communicate in assigned time slot channels, giving the user a continuous virtual communication path over the network. Communication systems providing a dedicated channel are known as "circuit-switched" networks. By using time division multiple access the need for a central hardware switch is effectively eliminated. The only central processing to be done on such a network is word slot assignments.

SECTION FOUR: DESIGN CONSIDERATIONS

The previous sections have outlined the overall structure and function of the experimental network´s multiplexers and demultiplexers. This section will outline the specific requirements placed on these devices.

As previously mentioned, each multiplexer and demultiplexer must be capable of providing first-in/first-out buffering and be able to store 192 bits of data (one 96 bit word ready for retransmission, the other 96-bit word for current data collection). Ideally, data input and data output would occur simultaneously. Unfortunately, the current state of the art has not yet progressed to the point where this is possible. The compromise is to interleave the READ and WRITE operations. The consequence of this, however, is that the rate at which information is written to or read from the buffer must be double that of the fastest input or output speed. Since each of the input and output speeds of these devices must be capable of handling bit rates up to 100 Mb/s, the buffer itself must be capable of handling data at 200 Mb/s. The result is a 192-bit first-in/first-out memory that can run at 200 million bits per second.

The fastest available FIFOs are TTL and operate one order of magnitude slowlier than the speed required. To meet the requirements it was necessary to use a faster family of logic. The fastest commercially available memories are emitter coupled logic (ECL) random access memories (RAMs). Since the ECL family does not have FIFOs, it was necessary to create one using these RAMs and various other ECL components.

The available ECL RAMs are limited to 30 Mb/s. Thus it was still necessary to slow down the effective bit rate through the memory without sacrificing performance of the overall multiplexer/demultiplexer. By converting the serial input data stream into 8-bit bytes that can be processed in parallel, it is possible to slow the data stream from 100 Mb/s to 12 megabytes/s. The memory unit configured as a FIFO would then only be required to run at 25 megabytes/s, which is within the range of the ECL memories.

Creating this very fast FIFO device was not a trivial matter. Serial to parallel conversions have to be performed in less than 10 nanoseconds (ns). After each serial to parallel conversion, data would have 80 ns to be absorbed into the memory (WRITE operation) before the next conversion occurred. The absorption process takes at least 30 ns. At the other end of the multiplexer/demultiplexer, the stored bytes of data are converted into a serial bit stream. After each parallel to serial conversion, data would also have 80 ns to be released from the memory (READ operation) before the next conversion occurred. This release process takes 30 ns as well. Thus careful synchronization is required to assure that during each

80 ns period both a complete WRITE and a complete READ operation can occur. Since each operation takes 30 ns, this leaves only a 20 ns margin. Most of this margin must be used for memory management overhead (incrementing registers, applying correct register contents to the RAM, etc.). Most of this project was involved in optimizing the hardware design such that it would meet the requirements listed above. In the following sections the resulting design is described.

SECTION FIVE: STRUCTURE OF A MULTIPLEXER

As previously outlined, the multiplexer performs the function of gathering digital information from the user at speeds less than or equal to 100 Mb/s and retransmitting the data in 100 Mb/s bursts. Figure 4 shows, in general, how the multiplexer and demultiplexer interact with the rest of the network. Each multiplexer receives a transmit clock signal used to synchronize data transmissions. This same clock signal is used to drive the rest of the multiplexer. For simplicity in this experimental network, the user clock signal was derived from the transmit clock signal by using a binary counter. Thus the frequencies available for the user clock would be equal to the transmit clock frequencies divided by 2^n $(n=0,...,7)$. Using information provided by the controller, the multiplexer can be programmed for user frequency selection as well as word slot assignments. The word slot assignments are used to determine when to transmit the 96-bit burst of data onto the "transmit data" line.

Figure 5 shows an overall diagram of the multiplexer. Using the "User Clock" line, the user synchronously enters his data into the multiplexer.

As previously mentioned, this data can be at speeds up to 100 Mb/s, requiring serial to parallel conversion. After every 8 bits are shifted in, the serial to parallel converter latches onto this byte for transfer to the memory unit. Meanwhile, more serial data can be entered in. The same signal that caused the byte to be latched is also used to alert the memory control unit of a "request to load" data into the RAM. This activates a 40 ns WRITE sequence during which the "input pointer" is applied to the RAM address lines and a "load-data" signal is sent to the RAM. This process is repeated until at least 12 bytes of data (a complete word) are stored.

Data is then clocked out to the network via a shift register in the parallel to serial converter. When the last bit of a byte is shifted out, the next byte of data is quickly loaded into the shift register from a nearby latch. The signal used to load the shift register is sent to the memory control unit over the "data request" line. This begins a 40 ns MEMORY READ sequence to provide the parallel to serial converter with a fresh byte of data. This sequence must 1) increment the "output pointer", 2) apply it to the address lines of the RAM, and 3) alert the parallel to serial converter (over the "latch data" line) that output of the RAM is the next byte of data. This process would repeat itself 12 times for every word slot being transmitted into.

This section provided an overall look at the data flow and structure of the multiplexer. In the next section, specific design considerations will be reviewed.

SECTION SIX: SPECIFIC DESIGN CONSIDERATIONS

This multiplexer was designed to function as a component in a lightwave communication experiment currently in progress at Bell Laboratories Crawford Hill Laboratory. This particular network operates at a speed of 16 Mb/s instead of 100 Mb/s. This experimental network already has microprocessor-controlled station units that can communicate with a central processor that keeps track of word slot assignments. Since the station microprocessor controllers are mounted in INTEL card cages, it was decided that it would be best to mount the multiplexer/demultiplexer on a card compatible with this system. Through the INTEL multibus, it is possible for control signals to be sent from the microprocessor station unit to the multiplexer/demultiplexer card. In this way, the multiplexer's word control unit and user clock generator can be programmed (see Figures 5 and 6). Namely, the microprocessor sends 3 bits to the user clock generator to specify which of the 7 frequencies the user has requested. 128 bits are sent to the word control unit, each bit representing whether or not permission has been granted for use of one of the 128 time slots. This programming is done synchronously with the microprocessor, and asynchronously with the 100 Mb/s network. Since the INTEL data bus transmitted TTL levels, it was necessary to provide a buffering and translation layer between my ECL circuit and the bus.

For ease of design, I decided to construct the circuit entirely out of ECL components. Although this was perhaps not the most efficient way, it did seem reasonable for such a high speed experiment.

To test the multiplexer, clock and data signals were provided from a Hewlett-Packard bit rate generator with ECL output levels. Figure 6 provides a more detailed view of the multiplexer structure. The clock from the bit generator entered the circuit on the "transmit clock" line to provide synchronization information for the entire circuit. The user clock is generated by applying the transmit clock as input to a 7 bit counter. The outputs of the counter, along with the transmit clock, are presented to an 8-to-1 multiplexer. The 3-bit number previously sent to the user clock generator is used to select the appropriate user clock frequency. The chosen user clock frequency is provided to the user to synchronously enter his data into the multiplexer. The same signal is then used to shift in the user´s data through an 8-bit shift register. A divide-by-8 counter keeps track of how many bits have been shifted in such that after every 8 bits, the "carry-out" signal is sent to latch the current 8 bits from the shift register.

The timing of this step is especially important since the latch will have as little as 10 ns to load the data in the shift register. The divide-by-8 "carry-out" provides the "load-request" signal shown in Figure 5. In the memory control portion, this signal is used to increment the counter containing the input pointer and set an RS flip-flop. This pulse continues down a 30 ns synchronous delay line while the output of the RS flip-flop causes the input pointer to be applied to the RAM address leads via a multiplexer. The RS flip-flop also enables the output of the delay line to be fed to the WRITE/ENABLE lead of the RAM. This entire process takes 40 ns, which is the maximum allowable time as described in Section 4. The process described above continues repetitively, filling up the RAM with data.

Meanwhile, the transmit clock is applied to a divide-by-96 counter that keeps track of the current bit number in any given word. At the end of each word, a "carry-out" is generated (called "word clock"), and applied to the word control unit. This word clock causes the word control unit to send the next WORD/ENABLE signal to an AND gate. If the WORD/ENABLE is true (that is, the upcoming word is to be used by the multiplexer), the transmit clock (attached to the other input of the AND gate) will be passed through to the output shift register.

Due to the fact that the word control unit must quickly generate a fresh WORD/ENABLE, the word control unit uses an internal look-ahead feature (see Figure 7). As a consequence, when the word clock enters the unit, it causes the already waiting WORD/ENABLE signal to pass through a flip-flop. Once this is complete, the next WORD/ENABLE is extracted from the memory in the following manner: the 7-bit counter that keeps track of the next word number is incremented. The four most significant bits select the appropriate 8 bits from a 16 X 8 bit RAM containing all WORD/ENABLEs. These 8 bits are applied to the input of an 8-to-1 multiplexer, which then selects the next WORD/ENABLE using the three last significant bits of the counter. This sort of architecture was chosen to minimize the time required to program the word control unit with WORD/ENABLEs. This way, the microprocessor station unit can completely program this unit using 16 (8-bit) data transfers instead of 128 (1-bit) data transfers.

Now, back to the shift register that is transmitting serial data into the network. Once the data in the 8-bit shift register has been emptied, the associated divide-by-8 counter will quickly reload the shift register from the associated latch. This operation is critical because it must be performed within 10 ns. The "carry-out" of the divide-by-8 counter (referred to in Figure 5 as "data request") is used to increment the register containing the output pointer and resetting an RS flip-flop. As with the "load request" signal, this pulse continues down a 30 ns synchronous delay line while the output of the RS flip-flop directs the output pointer to be applied to the address leads of the RAM through the multiplexer. When the "data request" signal emerges from the synchronous delay line, the output of the RS flip-flop causes this pulse to return to the latch to load the next byte of data. This entire process takes 40 ns, which again is the maximum allowable time as described in Section 4. This process of transferring data out of the multiplexer continues as long as an assigned word slot is in progress.

The processes of entering and retrieving data from the multiplexer have now been described. In order for these processes not to collide, careful synchronization of the request signals to the memory controller is required. This is achieved by initializing the state of all the counters in the circuit on a common "frame pulse" at the beginning of the frame (see Figures 6 and 7). Since the entire circuit is synchronous with the "transmit clock" (or a phased derivative of it) it is possible to assure that the "load-" and "data-requests" from either divide-by-8 counter will be separated by at least 40 ns. In this way we were able to implement all the design criteria outlined in Section 4.

SECTION SEVEN: STRUCTURE OF A DEMULTIPLEXER

The design criteria outlined in Section 4 applies as well to the demultiplexer. For this reason the topology of the demultiplexing circuit is very similar to that of the multiplexer (see Figure 8). One major difference between them, however, is that the input data will now be coming from the network, requiring that the serial to parallel converter be controlled by the word control unit instead of the user clock generator. The other major difference is analogous: the output data provided to the user through the parallel to serial converter must be controlled by the user clock generator. The rest of the demultiplexer circuit remains intact.

Thus, the word control unit keeps track of the current word in progress, and, during the proper word slot, causes the serial to parallel converter to shift in the 96-bit burst of data currently on the network. Memory "load-requests" are handled exactly as they were in the multiplexer. The user data is shifted out through the parallel to serial converter at the frequency provided by the user clock generator. Similarly, parallel to serial converter requests to memory are handled as in Section 5. Synchronization issues are just as important, and are handled in exactly the same manner. Therefore, it is possible to create a demultiplexer by using the same functional blocks as for the multiplexer, with only minor rearrangements.

SECTION EIGHT: RESULTS AND CONCLUSIONS

The total design of the multiplexer described in Section 5 was implemented in hardware at the Crawford Hill Laboratory of Bell Laboratories in Holmdel, New Jersey. The chip count was 40 ECL DIP and 6 TTL DIPs.

Mounted on a PC card compatible with the INTEL card cages, this multiplexer was tested with a microprocessor driven station card. It was found that control signals from the microprocessor could successfully load information into both the word control unit and the user clock generator. Using a Hewlett-Packard bit rate generator Model 8081A, serial to parallel conversion and storage could be fully achieved. Due to lack of time, synchronization bugs were not able to be completely worked out. The board is currently undergoing further testing and modification at Bell Laboratories.

Although time did not allow for the actual construction of a demultiplexer, the author feels that the design is similar enough to the multiplexer that the successful results from the multiplexer board test can be taken demonstration of demultiplexer design feasibility.

From the success that was achieved in the laboratory using commercially available hardware, the author concludes that some of this ideas may be implemented in the future using VLSI technology.

APPENDIX A: SCHEDULING ALGORITHM

The scheduling algorithm mentioned in Section 3 is presented here. This algorithm will minimize "blocking" in the network given that

1) once an assignment is made it will remain unchanged until user no longer needs the channel, and

2) network has no a priori knowledge of holding times for users.

The need for such an algorithm was to provide efficient utilization of the available bandwidth given the memory constraints of both the multiplexer and demultiplexer.

More specifically, since the multiplexer/-demultiplexer can only store two 96 bit words of data, it was necessary to assure that a time slot would be made available after each 96 bit word was collected/distributed. Thus, the maximum allowable wait between the conclusion of a 96-bit word collection/distribution and the next assigned time slot is the time required by the multiplexer/demultiplexer to collect/distribute the next 96 bits.

For scheduling purposes, the frame is then uniformly divided into $128/2^{n_o}$ word groups where 2^{n_o} is the number of time slots per frame to be assigned to a multiplexer/demultiplexer (n_o=0,1,...,7). By assigning one time slot form each group, it is assured that the aforementioned requirements will be satisfied.

If all the users required the same bandwidth, it would be possible to sequentially assign time slots in each "group" until there were no more available, and no "blocking" would occur. Unfortunately, this is not the case. Since the higher bandwidth users have more smaller size "groups" to choose from, it would be very easy to block a high bandwidth user if the sequential word slot assignment method was used.

For example, if two 1536 Kb/sec users (call them A and B) each requested two time slots per frame and then a 49 Mb/sec user (call him C) requested 64 times slots, the following scenario would occur.

> User A presents his request for 2 time slots. The 128 word frame (words #0-127) would be divided up into two groups, 64 words in each. Since no other words have been assigned, user A would be assigned time slots #0 and #64. Similarly user B would be assinged the NEXT available time slots in each group (#1 and #65). Then user C presents his request for 64 time slots. The frame would then be divided up into sixty-four groups, 2 words in each (that is, #0 and #1, #2+3,..., #64+65,..., #126+127). Since two

of these word groups have already been assigned (#0+1, #64+65), user C will be "BLOCKED" from using the network, in spite of the fact that there are 124 available time slots.

The basic cause for this "blocking" was due to the fact that sequential scheduling does <u>NOT</u> evenly distribute word slot assignments throughout a frame.

For this reason, an extension of the previous idea of uniform division is proposed. Instead of sequentially assigning a time slot is a given "word group", assign the word to the section of the group that currently has the <u>lowest</u> assignment density. The procedure would work as follows:

1) divide the "word group" of interest in half.

2) examine each half to find the assignment density $\left|\frac{\text{\# assigned}}{\text{\# in each half}}\right|$

3) The time slot will then be assigned to the half of the word group with the lowest assignment density (if the densities are equal, choice is arbitrary).

4) Repeat steps 1, 2 and 3 until the chosen "half" is only one word, at which time the assignment can be made.

In order to demonstrate this procedure, the previous scenario will be re-enacted

User A presents a request for 2 time slots. The frame is divided into two groups (WORDS #0-63 and #64-127). the first group is now subdivided into equal parts (#0-31 and #32-63) and the assignment densities are examined. Since we assume that there were no previous assignments, the assignment densities are both zero. For this example, the arbitrary choice will always be the lower numbered word group. After this procedure is repeated several times, time slot #0 is selected. This entire process is repeated for the other word group (#64-127) until slot #64 is selected. Note that up to this point, the assignments made were identical to the "sequential" example.

When user B requests 2 time slots, the same two groups of 64 words are chosen (#0-63 and #64-127). After the first subdivision of the #0-63 word group is made, it is noted that the #0-31 subgroup has a higher assignment density $\left|\frac{1}{32}\right|$ than the #32-63 subgroup density $\left|\frac{0}{32}\right|$. This means that the assignment will fall into the second subgroup. Continuing the procedure, time slot #32 will be chosen for the first word and time slot #96 will be chosen for the second. Note at this point, that the total assignment density is uniform over the entire frame (#0,37,64,96).

When the 49 Mb/sec user (C) asks for 64 time slots, the frame is divided into sixty-four groups of two (#0+1, 2+3,..., 32+33,..., 64+65,..., 96+97,..., 126+127). Note now that in each word group, there is at least <u>one</u> unassigned time slot available. Following the algorithm completely through, user will be assigned all the even numbered time slots except for #0, 32, 64 and 96. These four will be replaced by #1, 33, 65, and 97.

If the users cancel service in the same order it is asked for, this algorithm will allow for 100% utilization of the available bandwidth, since the assignment distribution will always be uniform. If this is not the case, it will not be possible to maintain a uniform assignment distribution, and blockage can occur even if sufficient bandwidth is available. The only solution to this problem is to dynamically re-schedule <u>all</u> of the assignments after each user cancels his service. This solution would be difficult to implement in hardware since <u>the entire network</u> would need to be re-programmed at the same instant in time (between 2 frames).

Given that such dynamic re-scheduling is not an immediate possibility, the proposed algorithm will provide for maximum utilization of the network bandwidth given no knowledge a priori of future network requirements or conditions.

<u>REFERENCE</u>

1. E. G. Rawson, "Application of Fiber Optics to Local Networks," Proceeding of the LACN Symposium, 155-167, May, 1979.

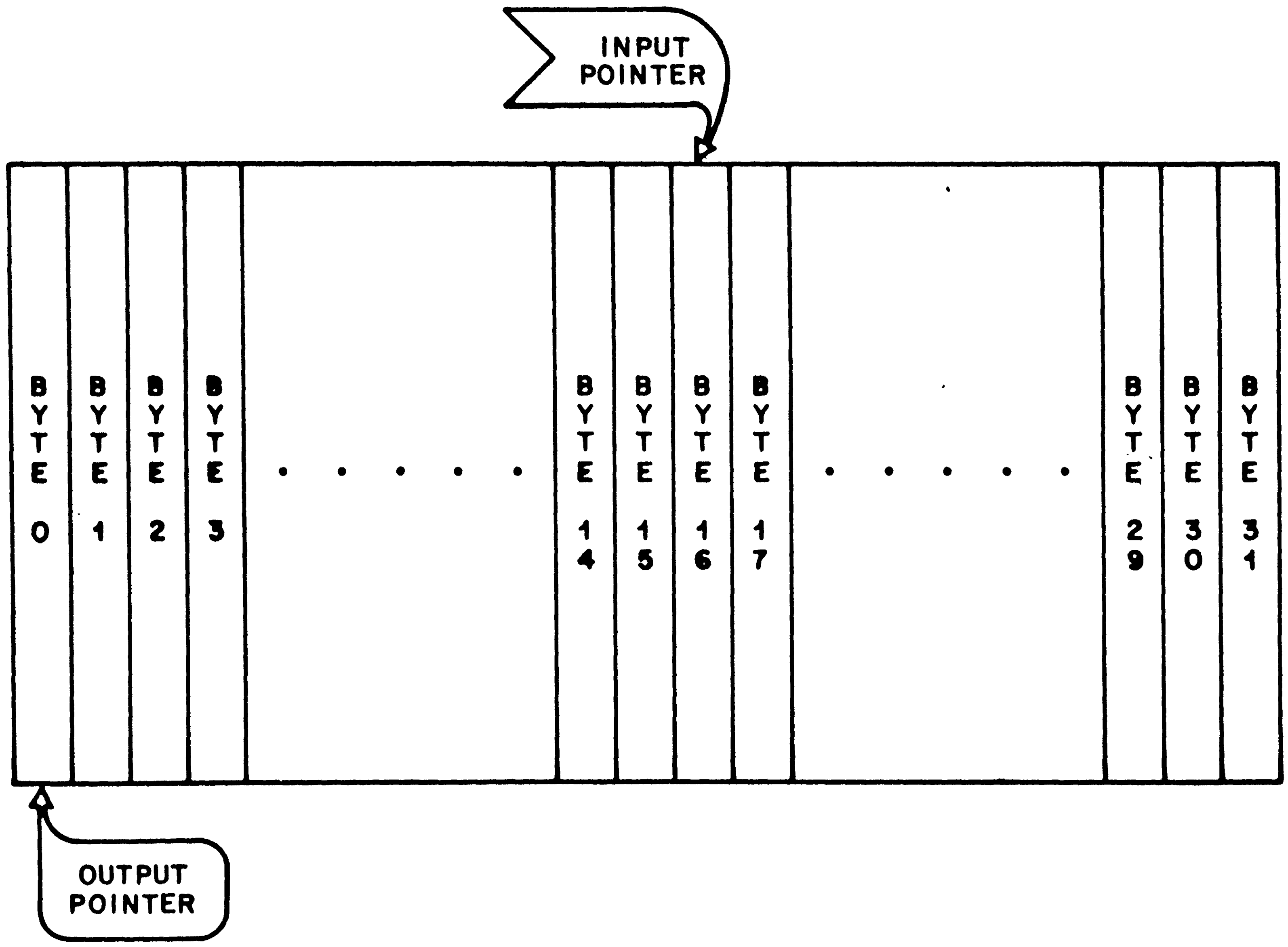

FIGURE 1 MEMORY STACK

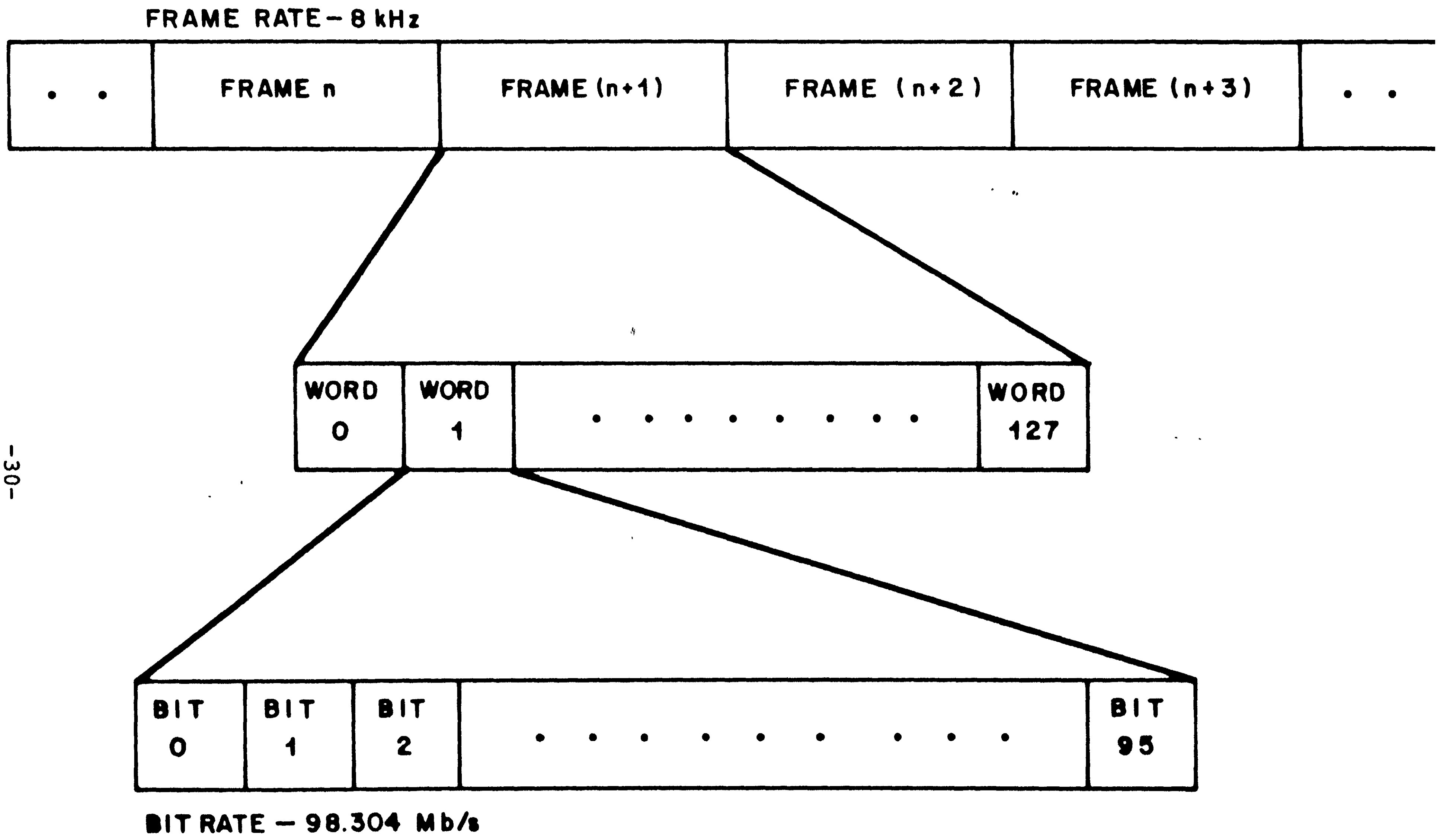
FRAME RATE – 8 kHz
. .
FRAME n
FRAME (n+1)
FRAME (n+2)
FRAME (n+3)
. .
WORD 0
WORD 1
. . . .
WORD 127
BIT 0
BIT 1
BIT 2
. . . .
BIT 95
BIT RATE – 98.304 Mb/s

FIGURE 2

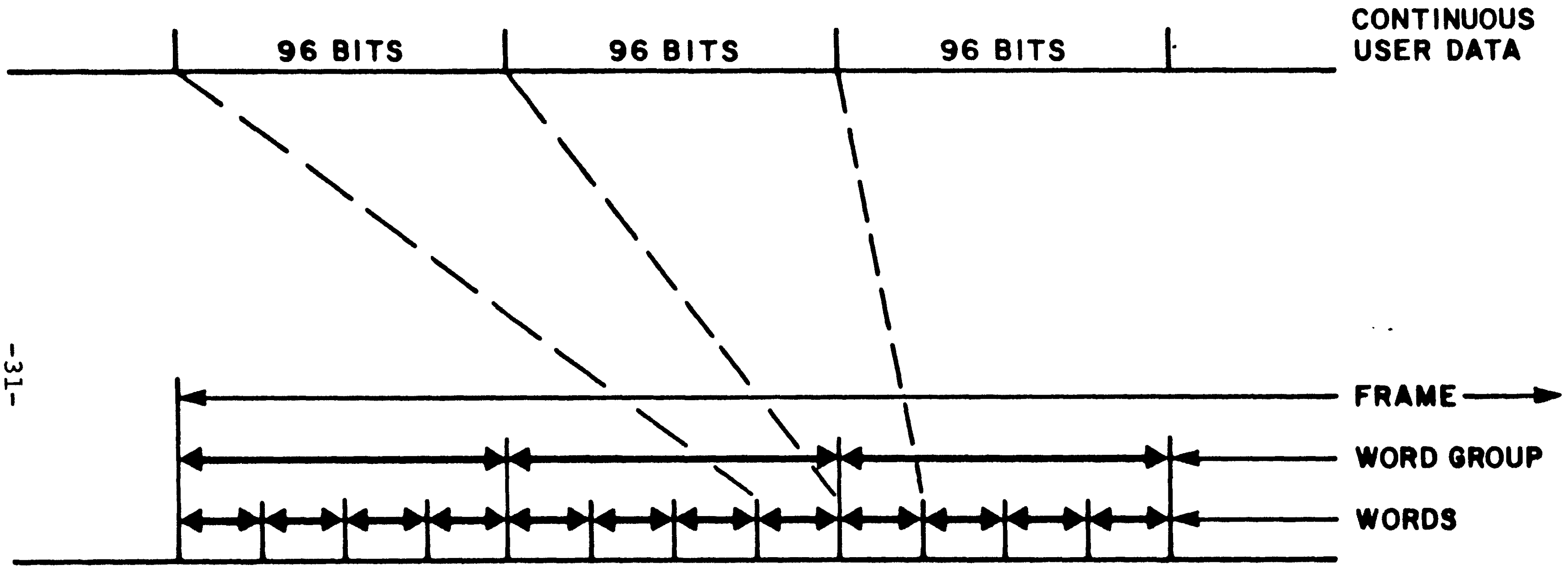

FIGURE 3 EXAMPLE: 25 Mb/s USER

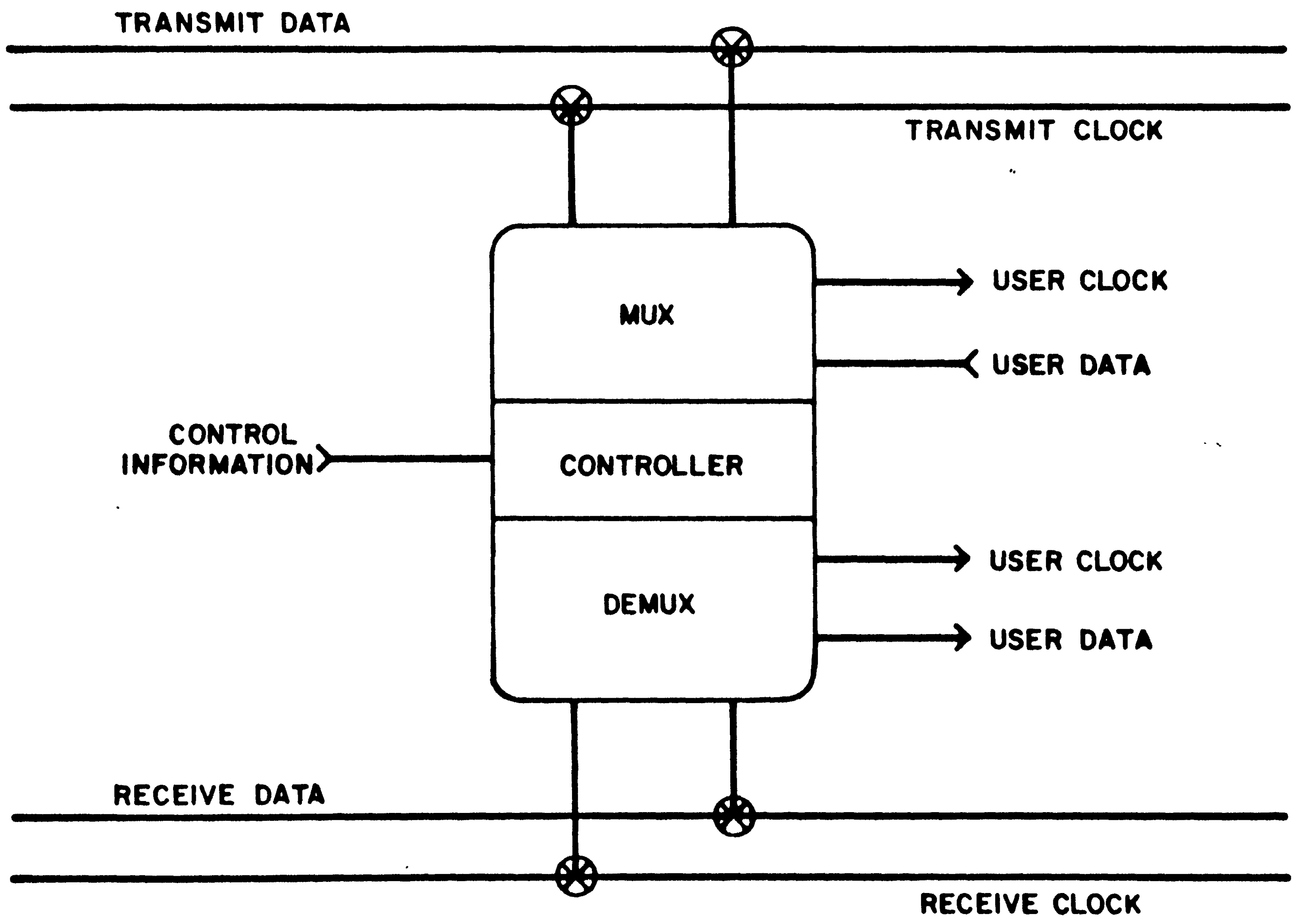
TRANSMIT DATA
TRANSMIT CLOCK
MUX
CONTROLLER
DEMUX
USER CLOCK
USER DATA
USER CLOCK
USER DATA
CONTROL INFORMATION
RECEIVE DATA
RECEIVE CLOCK

FIGURE 4

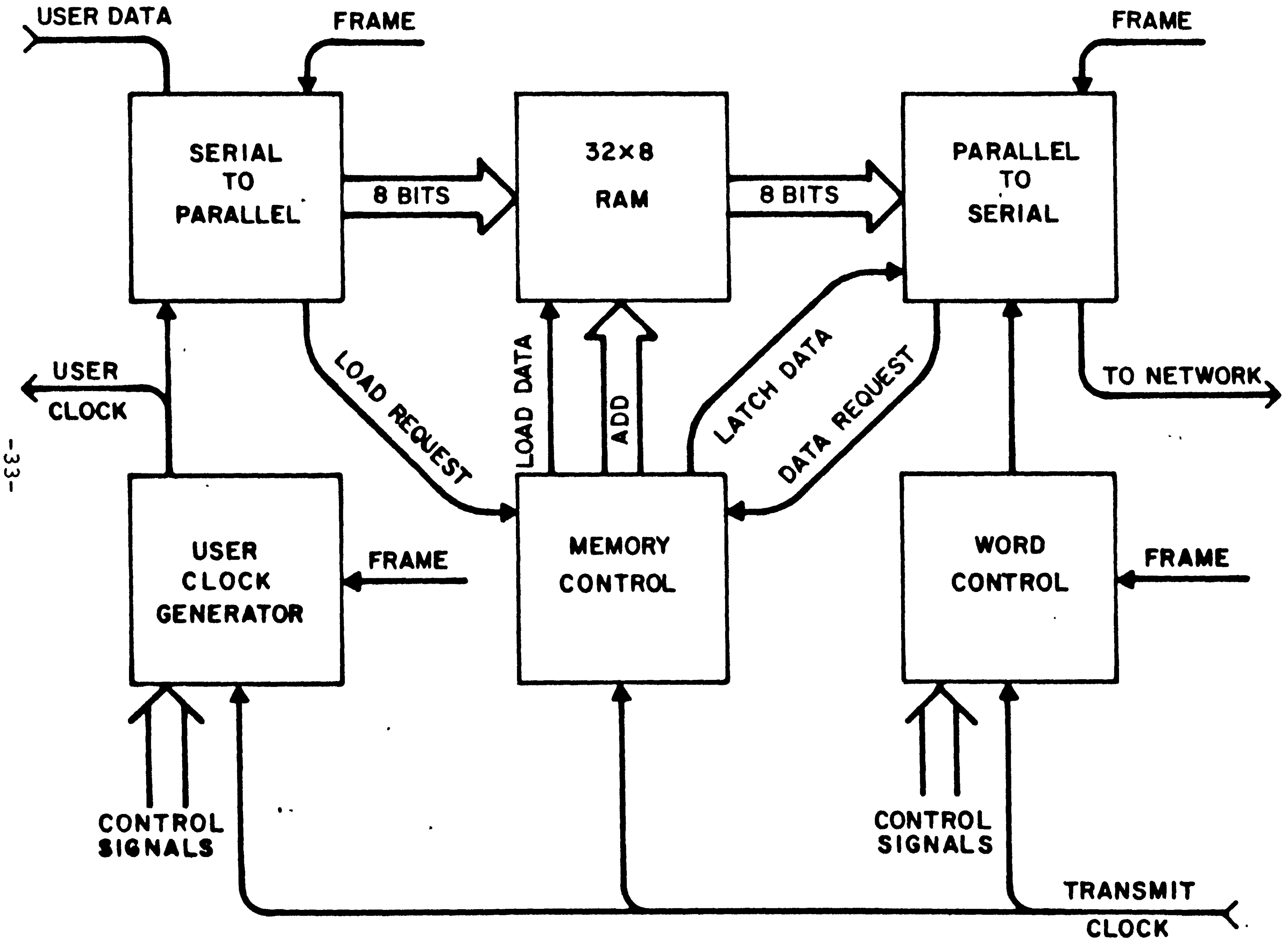

FIGURE 5 MULTIPLEXER

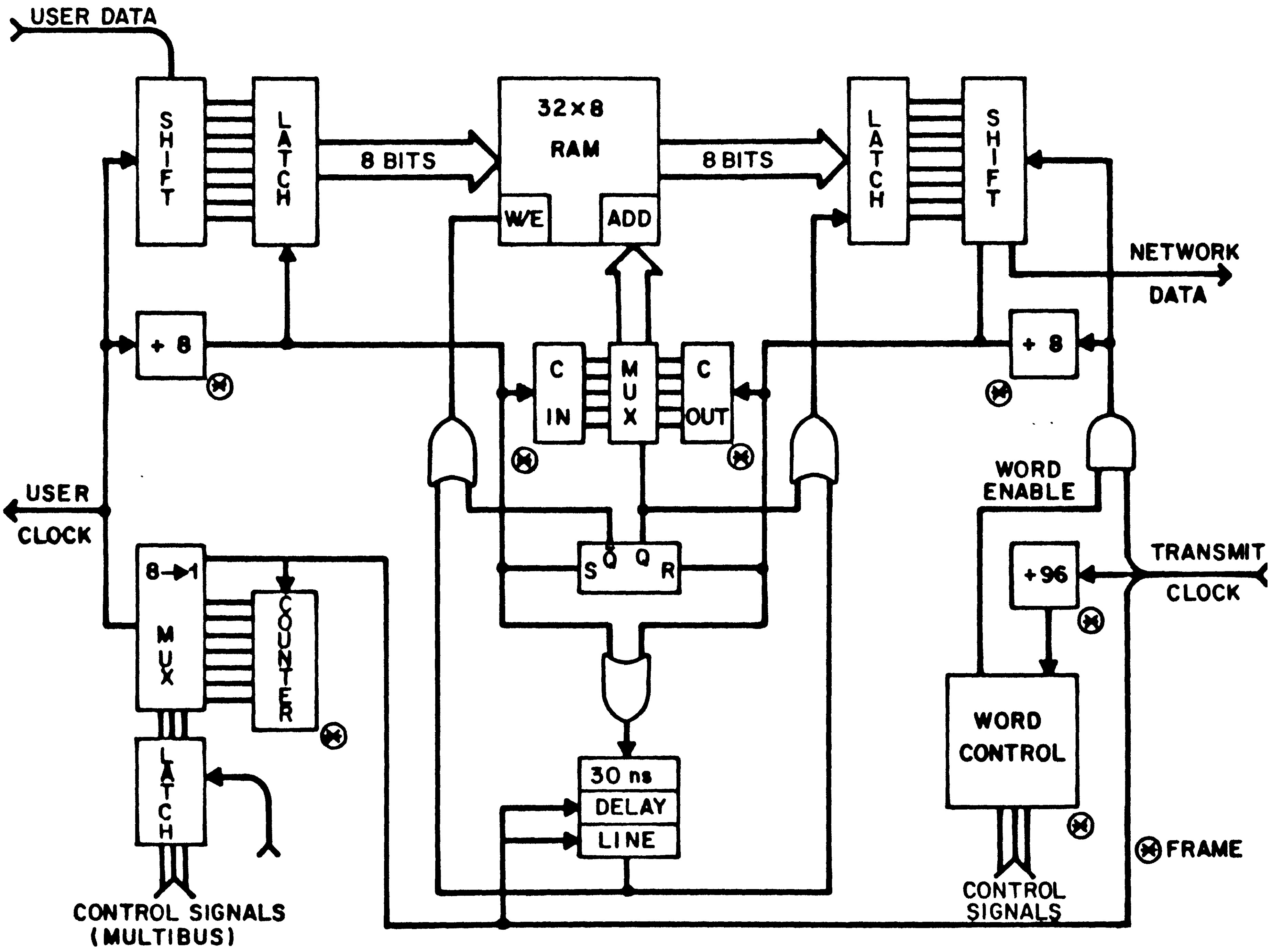
USER DATA
SHIFT
LATCH
8 BITS
32×8
RAM
W/E
ADD
8 BITS
LATCH
SHIFT
NETWORK
DATA
+ 8
C
IN
MUX
C
OUT
+ 8
WORD
ENABLE
USER
CLOCK
8→1
MUX
COUNTER
S
Q̄
Q
R
TRANSMIT
CLOCK
+96
WORD
CONTROL
30 ns
DELAY
LINE
LATCH
CONTROL SIGNALS
(MULTIBUS)
CONTROL
SIGNALS
FRAME

FIGURE 6

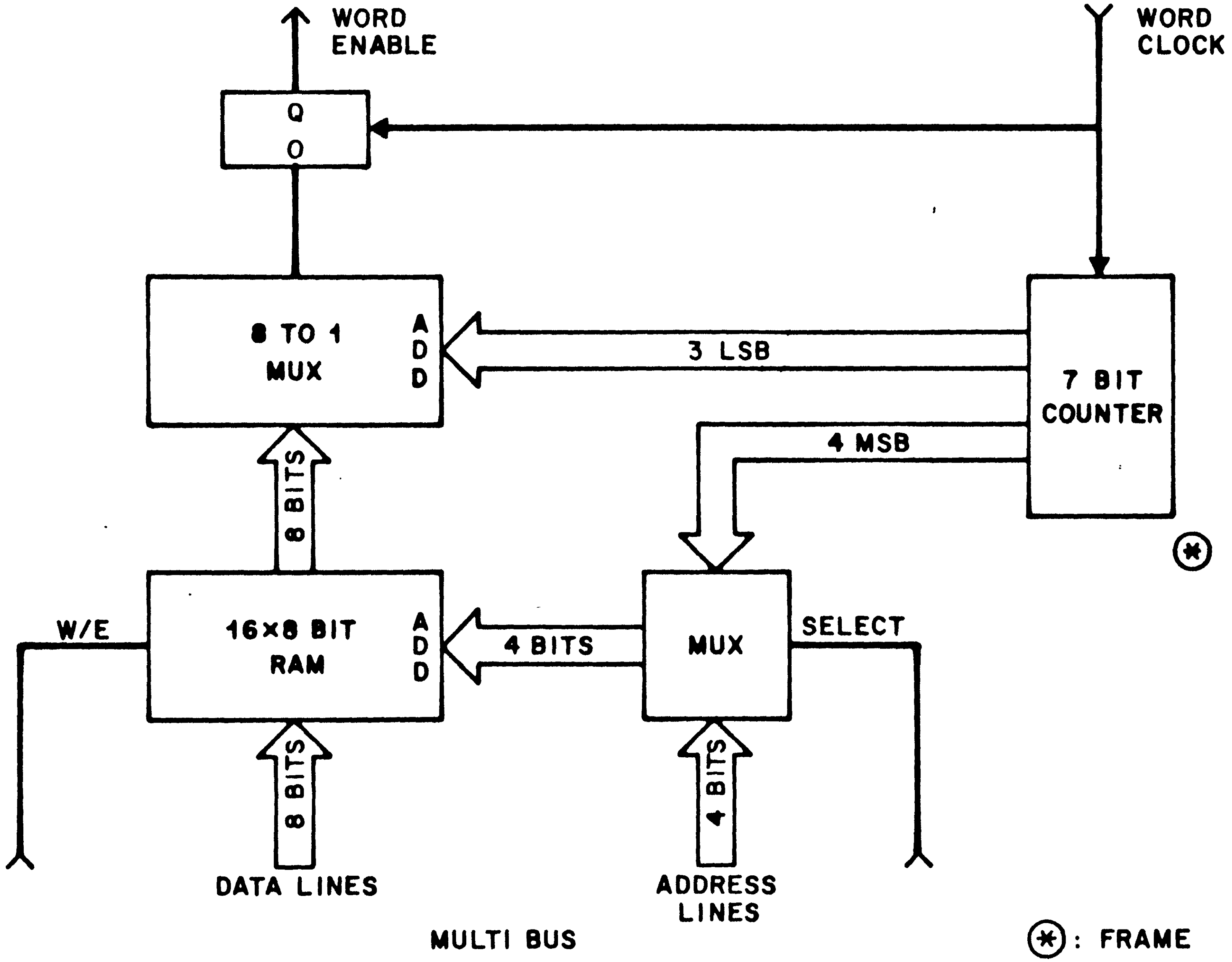

FIGURE 7 WORD CONTROL

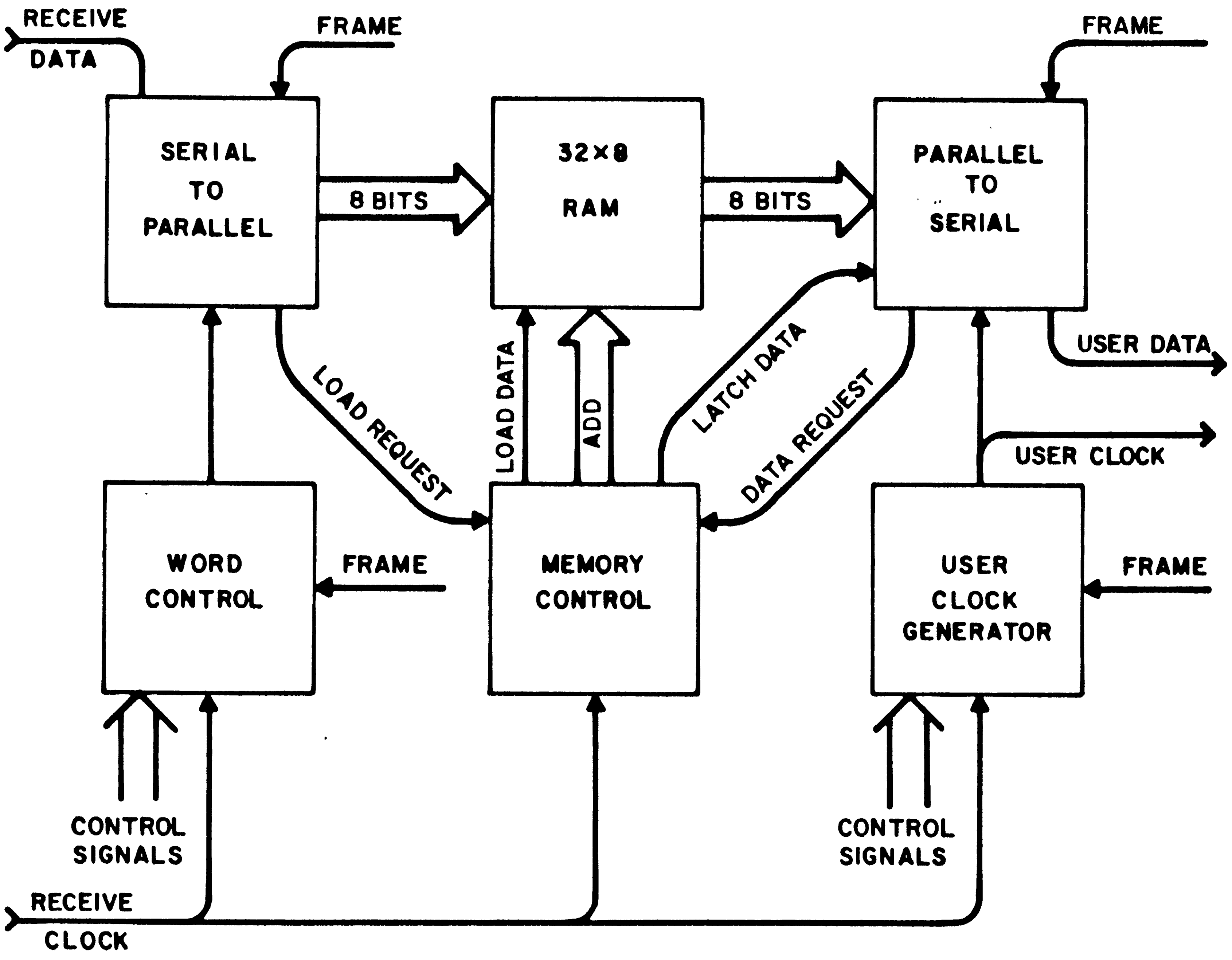

FIGURE 8 DEMULTIPLEXER