General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

NASA TECHNICAL MEMORANDUM

NASA TM X-62,443

NASA TM X-62,443

QUASI-PERFECT FIFO – SYNCHRONOUS OR ASYNCHRONOUS WITH APPLICATION IN CONTROLLER DESIGN FOR THE UNICON LASER MEMORY

Raymond S. Lim

Ames Research Center
Moffett Field, Calif. 94035

January 1974

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
TM X-62,443		

4. Title and Subtitle	5. Report Date
QUASI-PERFECT FIFO – SYNCHRONOUS OR ASYNCHRONOUS WITH APPLICATION IN CONTROLLER DESIGN FOR THE UNICON LASER MEMORY	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Raymond S. Lim	A-6091
	10. Work Unit No. 997-36-60-09-08
9. Performing Organization Name and Address	11. Contract or Grant No.
Ames Research Center, NASA Moffett Field, Calif. 94035	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Memorandum
National Aeronautics and Space Administration Washington, D. C., 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The FIFO, a first-in-first-out memory buffer, is an elastic digital memory. Its main application is in data buffering between devices operating at different rates. Data written into the top is moved autonomously down toward the bottom of the FIFO to the lowest unoccupied location. Data read from the bottom of the FIFO will cause data from the top to move autonomously down toward the bottom. The concept of the FIFO just described is a simple one, but its implementation is very complex. Currently, FIFO is available in MOS LSI asynchronous form with data rate in the 1 MHz region. The FIFO described in this paper yields a simple high-speed iterative implementation, either synchronous or asynchronous. Because of this simple iterative structure, the FIFO is easily expandable in both number of words and bits per word, and it is very attractive from the viewpoint of integrated-circuit production. For these reasons, this FIFO is called the quasi-perfect FIFO. For the synchronous FIFO, a model was built and successfully used in the Controller for the UNICON Laser Memory. For the asynchronous FIFO, a model was built and also successfully used in a high-performance magnetic tape Controller.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
FIFO Elastic memory Buffer memory	Unclassified – Unlimited STAR Category 62

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	15	$3.25

*For sale by the National Technical Information Service, Springfield, Virginia 22151

QUASI-PERFECT FIFO – SYNCHRONOUS OR ASYNCHRONOUS

WITH APPLICATION IN CONTROLLER DESIGN FOR THE UNICON LASER MEMORY

Raymond S. Lim

Institute for Advanced Computation

Ames Research Center, NASA
Moffett Field, Calif. 94035

SUMMARY

The FIFO, a first-in-first-out memory buffer, is an elastic digital memory. Its main application is in data buffering between devices operating at different rates. Data written into the top is moved autonomously down toward the bottom of the FIFO to the lowest unoccupied location. Data read from the bottom of the FIFO will cause data from the top to move autonomously down toward the bottom. The concept of the FIFO just described is a simple one, but its implementation is very complex. Currently, FIFO is available in MOS LSI asynchronous form with data rate in the 1 MHz region. The FIFO described in this paper yields a simple high-speed iterative implementation, either synchronous or asynchronous. Because of this simple iterative structure, the FIFO is easily expandable in both number of words and bits per word, and it is very attractive from the viewpoint of integrated-circuit production. For these reasons, this FIFO is called the quasi-perfect FIFO. For the synchronous FIFO, a model was built and successfully used in the Controller for the UNICON Laser Memory. For the asynchronous FIFO, a model was built and also successfully used in a high-performance magnetic tape Controller.

INTRODUCTION

The FIFO, a first-in-first-out memory, is an elastic two-port digital data buffering device. The designation of a FIFO is usually in the notation of n-bit $\times$ m-location. From top to bottom, the FIFO is numbered as locations 0, 1, 2, . . . , $m - 1$ in that order. Data input to the FIFO can only be written into the top, and data output from the FIFO can only be read from the bottom. Both input and output can be accessed simultaneously. Data written into the top will automatically fall toward the bottom at some rate to the lowest unoccupied location. Data read from the bottom also will automatically cause data at the top to fall toward the bottom. Operational status indicators for the FIFO, such as top-empty, bottom-full, ¾-full, ¾-empty, etc., are usually provided.

The application of a FIFO ranges from slow data buffers for communication systems, queuers for peripheral resources management, to real-time high-speed data buffering of digital mass memories. At present, FIFOs are available as a MOS LSI chip with a maximum data transfer rate of about 1 MHz. The implementation of such a FIFO is the use of a random-access memory (RAM) with complex control logic consisting of read and write pointers. With such an implementation, the

FIFO is usually an asynchronous device; that is, the FIFO has no provision to be clocked by a system clock, whereas a system clock is usually used in the design of large digital systems. To the user, the implementation of the FIFO is transparent. Therefore, the user sees the FIFO as a two-port device. However, from the hardware viewpoint, the RAM-type implementation approach has three obvious difficulties. The first is that the speed is limited; second, the control is too complicated; and third, the device is t.. ay a one-port device since a RAM is used.

The implementation of the FIFO to be described in this paper will not use a RAM, but uses conventional registers. The control, instead of a complex sequential network using counters as pointers to control the read and write operations, is a simple iterative network. The network consists of identical sequential circuits (SC), one SC per each location in the FIFO. The SC consists of one flip-flop and a few gates. The state of the flip-flop also indicates whether that FIFO location has valid data or not. The implementation can be either synchronous or asynchronous. For the synchronous FIFO, the input and output data transfer rate is equal to one-half the system clock frequency. For the asynchronous FIFO, this transfer rate is equal to one-half the speed of the data registers used. For STTL registers, this speed is in the region of 20 ns.

PRELIMINARY DISCUSSION

The FIFO, in its most elementary form, can be described as a two-port network as shown in figure 1. The rules for data transfer into and out of the FIFO from and to an external device are as follows:

Input: Device inputs data into FIFO if and only if (iff) FIFO top is empty (FIFO top is not full).

Output: Device takes data from FIFO iff FIFO bottom is full.

The above rules for data transfer imply that some system of protocols with the FIFO is required. With respect to these data transfer protocols of the FIFO, the system designer must be concerned with data rate error (DRE) since the system cannot wait too long for the FIFO to become top-empty or bottom-full. A DRE can exist under two conditions:

Input: There exists a DRE if top of FIFO is full when the external device must write a word of real-time data into the FIFO.

Output: There exists a DRE if bottom of FIFO is empty when the external device demands a word (in a real-time sense) from the FIFO.

In order to prevent against possible DRE unreported, the FIFO statuses of top-empty and bottom-full are important to the system designer. If a DRE does occur as described above, such an error condition must be reported to the control computer. The statuses ¾-full and ¾-empty are also useful to the system designer if the FIFO is used to transfer data between a high-speed mass memory and a large central memory (CM) system (usually core or IC). In such a CM system, usually there are many devices and processors connected onto it. There exists frequent memory access conflicts between devices and processors. If the conflicts are high enough such that the FIFO becomes

¾-full (or ¾-empty dependent upon write or read mass memory), the system designer can use the ¾-full status to obtain a higher priority access into the CM system. In a sense, the FIFO provides the system designer with a look-ahead status report to prevent possible DRE.

DETAILED DISCUSSION

The FIFO, in its most elementary detailed block diagram form, is shown in figure 2. The FIFO shown is an n-bit X-4-location FIFO. Each location consists of a sequential-circuit (SC) and a data register (REG). All four locations are identical, thus the FIFO is an iterative sequential network. At the bottom of the FIFO, an optional holding-register (HR) is added. This HR addition is mainly for interface control. The statuses ¾-full and ¾-empty are formed by two combinational networks (CNET). The implementation of SC and the register at each location can be either clocked or unclocked. If clocked by a system clock, the FIFO is called a synchrnous FIFO, otherwise it is called asynchronous. For a 4-location FIFO, the locations are numbered sequentially as FIFO0, FIFO1, FIFO2, and FIFO3. Within the SC of each location, there is only one flip-flop (FF) to indicate the status of that location. The output of the FF is Q. A 4-stage FIFO would then have Q0, Q1, Q2, and Q3. A FIFO location full of valid data is indicated by Q = 1, otherwise it is empty.

The operating rules of the FIFO can now be stated:

Write: Write into FIFO0 iff $\overline{Q0} = 1$, meaning FIFO0 is empty, or not full.

Read: Read from FIFO3 iff $Q3 = 1$, meaning FIFO3 is full.

Transfer: FIFOi → FIFO$i + 1$ iff $Qi = 1$ and $\overline{Qi + 1} = 1$, meaning FIFOi is full and FIFO$i + 1$ is empty.

Output: First read FIFO3 into HR. Next gate data out from HR when convenient. A read of FIFO3 into HR will cause a transfer ripple up from FIFO3 to FIFO0.

Input: A successful write into FIFO0 will cause a transfer ripple down from FIFO3 to FIFO3.

¾ Full: ¾-Full = Q3 Q2 Q1

¾ Empty: ¾-Empty = $\overline{Q2}$ $\overline{Q1}$ $\overline{Q0}$

The write and read rules, as mentioned earlier, require some protocols with the FIFO. For example, consider a write mass memory (MM) operation where data flow is from CM → FIFO → MM. Because of the FIFO, the control logic that controls the data flow can be separated into two parts – the result is a simpler control for each part. One part is mainly concerned with data flow from CM into FIFO. The other part is concerned with data flow from FIFO into the mass memory. The control logic that brings data from CM → FIFO can make a protocol with top of FIFO. Therefore, DRE can never occur. The control logic that takes data from FIFO → MM cannot make use of protocol since MM is usually a real-time device. In this case, possible DRE must be appropriately indicated.

The interstage data transfer within the FIFO is automatic. For example, FIFO1 is automatically transferred to FIFO2 if FIFO1 is full and FIFO2 is empty. If, however, FIFO0 is also full, then FIFO0 → FIFO1 takes place immediately after FIFO1 → FIFO2 is finished.

SYNCHRONOUS IMPLEMENTATION

The synchronous implementation, as mentioned earlier, is to use a system clock to synchronize all logic elements that have memory within the FIFO. That is, all flip-flops and registers are clocked. Each location of the FIFO consists of an SC and a data register. The design of the register is a simple task. One simply chooses a synchronous register, such as the 74179 (or 74S179), and connects many of these registers in parallel until the desired number of data bits are obtained. The design of the SC can be obtained directly from the operating rules of the FIFO. The simplest design is shown in figure 3 where each SC consists of one JK flip-flop (JK-FF) and one AND gate. For SC1, it consists of FF F1 and AND gate G2. The JK-FF can be any type, however, a 74S112 was used successfully in the Unicon Controller. Note that the 74S112 is a negative-edge triggered FF. Consider FIFO stage 2 in figure 3, the input equations for the FF and the register are:

$LOAD2 = Q1\ \overline{Q2}$; strobe data from REG1 into REG2.

$J2 = LOAD2$; set input to JK FF.

$K2 = LOAD3$; reset input to JK FF.

$LOAD3 = Q2\ \overline{Q3}$; strobe data from REG2 into REG3.

The above equations are the same as the transfer rule stated earlier. For example, LOAD2 is the signal that strobes the data from REG1 into REG2. The signal LOAD2 = 1 iff Q1 = 1 and $\overline{Q2}$ = 1, meaning REG1 is full and REG2 is empty. Likewise, FF2 is set iff LOAD2 = 1, and reset iff data transfer from REG2 to REG3 is finished.

In order to further explain the FIFO operation, timing diagrams for write and read FIFO are shown in figures 4 and 5, respectively. For the synchronous implementation, note that the maximum data transfer rate for writing and reading the FIFO is equal to one-half the system clock frequency. However, internal data propagation from top to bottom of FIFO is equal to the system clock frequency.

ASYNCHRONOUS IMPLEMENTATION

The asynchronous implementation of the FIFO is slightly more complicated than the synchronous implementation. The detailed logic for a 3-stage asynchronous FIFO is shown in figure 6. The SC consists of one D-FF (7474), two AND gates, and one one-shot (OS, type Fairchild 9602). It should be noted that the D-FF is connected as a T-FF. The purpose of the OS is to provide a delay for the data register (74174). This delay must be greater than the sum of the data register set-up time plus propagation delay time. For a 74S174, this delay should be about 20 ns. Likewise, if a 74S174 is used for the data register, a 74S74 should be used for the D-FF. In this case, the speed of the FIFO is limited by the OS because most OSs have a minimum pulse width. For a F9602, this minimum pulse width is about 70 ns. The timing diagram for this FIFO is shown in figure 7.

APPLICATION IN UNICON CONTROLLER

The UNICON is a digital laser memory manufactured by Precision Instruments, Inc. It is part of the mass memory system associated with the ILLIAC-IV computer, and it has an on-line storage capacity of about 10^{12} bits. The UNICON is connected to the CM through a Controller [1]. The purpose of the Controller is to control the data flow between CM and UNICON. For controlling this data flow, the Controller was designed around the concept of a FIFO, which is used as an elastic memory buffer between CM and UNICON.

The Controller was designed and fabricated in 1973 by the Institute for Advanced Computation (IAC). Briefly, the Controller is a modern channel controller using the concept of swapping register sets and a minicomputer (a PDP-11) to control its internal memory management functions. For this reason, the minicomputer is called the UNICON Memory Management Processor (UMP). UMP obtains its control programs through its own virtual address hardware interface into the CM. By means of this virtual address space, UMP is capable of addressing up to 4096 pages (512W × 36B) in CM. For a write operation, data flows from CM to FIFO and then to UNICON, and vice versa for a read operation. As of this writing, error control in the data stream is in two parts. The controller performs only error detection by means of appending a 128-bit checksum at the end of a page of data, which is a single parity check symbol in $GF(2^{128})$. The UNICON performs both an error detection by means of the checksum and error correction by means of a shortened (80, 64) Fire Code for correcting all single-burst errors with burst length $b \leq 6$. The FIFO is included in the error detection data stream. A future plan for augmenting the error correction capability is to concatenate an outer code to the existing (80, 64) Fire Code [2].

As mentioned earlier, the design of the Controller for controlling data flow uses an FIFO in the data stream. A functional block diagram of this design is shown in figure 8. The size of the FIFO is chosen to be 16WX16B. The UNICON data rate is 3.2 μs per 16-bit word. The CM cycle time, including interface protocols, is 1.4 μs per 36-bit word. MDR is a 37-bit (36 data bits plus parity) data register interfaced with CM. It also performs the 36-bit to 16-bit word multiplexing. HR is a 16-bit register interfaced with UNICON. MISR is a status register containing a word-count and status indicators such as DRE, parity error, and checksum error. There are two separate control sections called CM-CONTROL and UNICON-CONTROL. These two control sections are totally isolated from each other with no controls in common. This is made possible by using the FIFO as a buffering device in the data stream. In this application, the UNICON can be considered as a 32-bit word memory device with a data transfer rate of 6.4 μs per word. For the chosen FIFO size of 16W × 16B, the FIFO offers an elastic buffering action of about 49 μs (8 × 6.4), which is 35 CM cycle times if there exists no memory access conflict in CM. So for all practical purposes, there should never be a DRE occurring during data transfer.

For a write operation, the FIFO initially is filled with data header information. CM-CONTROL is started to transfer data from CM to MDR, and then to the FIFO by making a protocol with the top of FIFO to ensure that data is deposited into the FIFO iff top of FIFO is empty. This process is repeated until the word count in MISR equals 512. Concurrently, UNICON-CONTROL is started to pull data off from the bottom of FIFO into HR, and then from HR to UNICON on a real-time demand basis. If the bottom of the FIFO is empty during this pulling, the DRE status in MISR will be set. This process continues until the UNICON has pulled 1024 16-bit words off the bottom of the FIFO. At any instant of time, the data flow within the FIFO is quite an interesting phenomenon. There may be data coming into the top and leaving the bottom of the FIFO

concurrently since it is truly a two-port device. As data is deposited into the top, it moves autonomously down towards the bottom to the lowest unoccupied location at the clock rate. As data is pulled off the bottom (location 15), this action causes the data in the next full location to move toward the bottom also at the clock rate. Thus, at any instant of time, it is possible that there may be more than one piece of data, starting at various locations within the FIFO, moving down toward the bottom of the FIFO concurrently.

For a read operation, the data flow is very similar to the write operation as just described. The differences are that data from the UNICON in real-time is deposited into the top of the FIFO through gate G1, and data to CM is taken off the bottom of the FIFO.

CONCLUSION

This paper has presented a design of the FIFO. The design uses registers for implementation, which is a deviation from the conventional approach of using random-access memories. The implementation can be either synchronous or asynchronous. The control logic for this FIFO design is a simple iterative network, using one flip-flop and one gate for each stage of the FIFO in the case of the synchronous implementation. The speed of the FIFO is limited only by the speed of the registers used. It is called the quasi-perfect FIFO because it has the characteristics of being simple in design, iterative in structure, high-speed, easy to produce from the IC viewpoint, and expansion capability in both words and bits per word. For the synchronous design, a model of the FIFO was built and successfully used in a controller for the UNICON laser memory. For the asynchronous design, a model of the FIFO was built and successfully used in a high-performance magnetic-tape controller for the Potter Instruments Model AT1802.

ACKNOWLEDGEMENT

The author wishes to thank Dr. Mel Pirtle for his suggestion in simplifying the design of the FIFO to its present form.

REFERENCES

1. Lim, R. S.: *A Channel Controller for the UNICON Laser Memory*. (To be submitted for publication as a TM.)

2. Lim, R. S.: *Augmented Burst-Error Correction for UNICON Laser Memory*. NASA TM X-62,442, 1975.

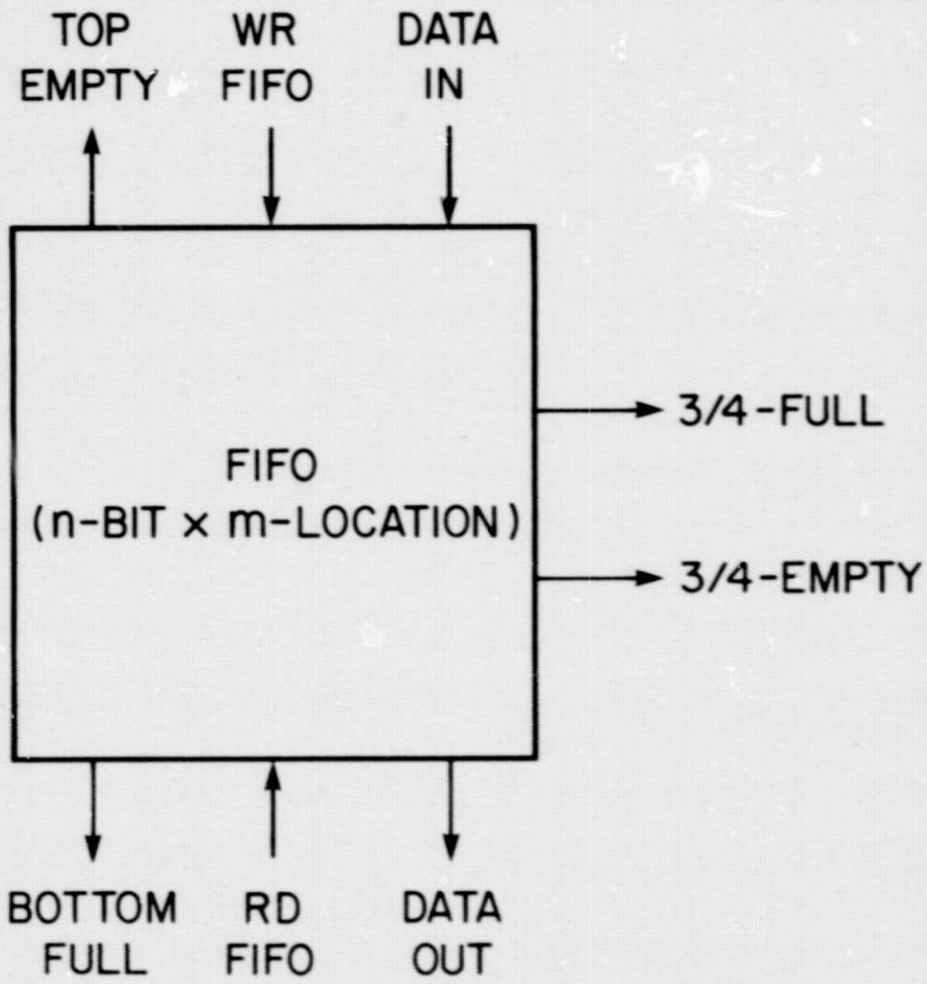

Figure 1.– FIFO representation in its most elementary form.

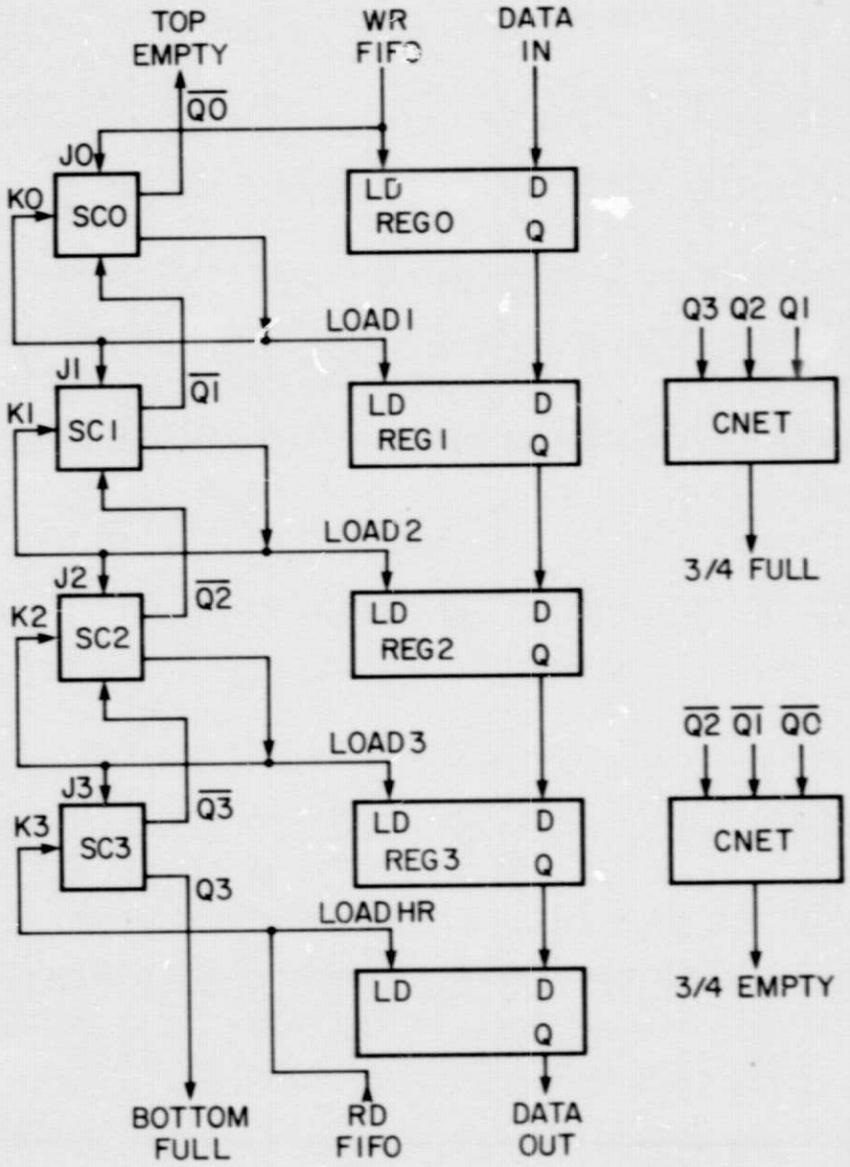

Figure 2.– Detailed block diagram of an n-bit X 4-location FIFO.

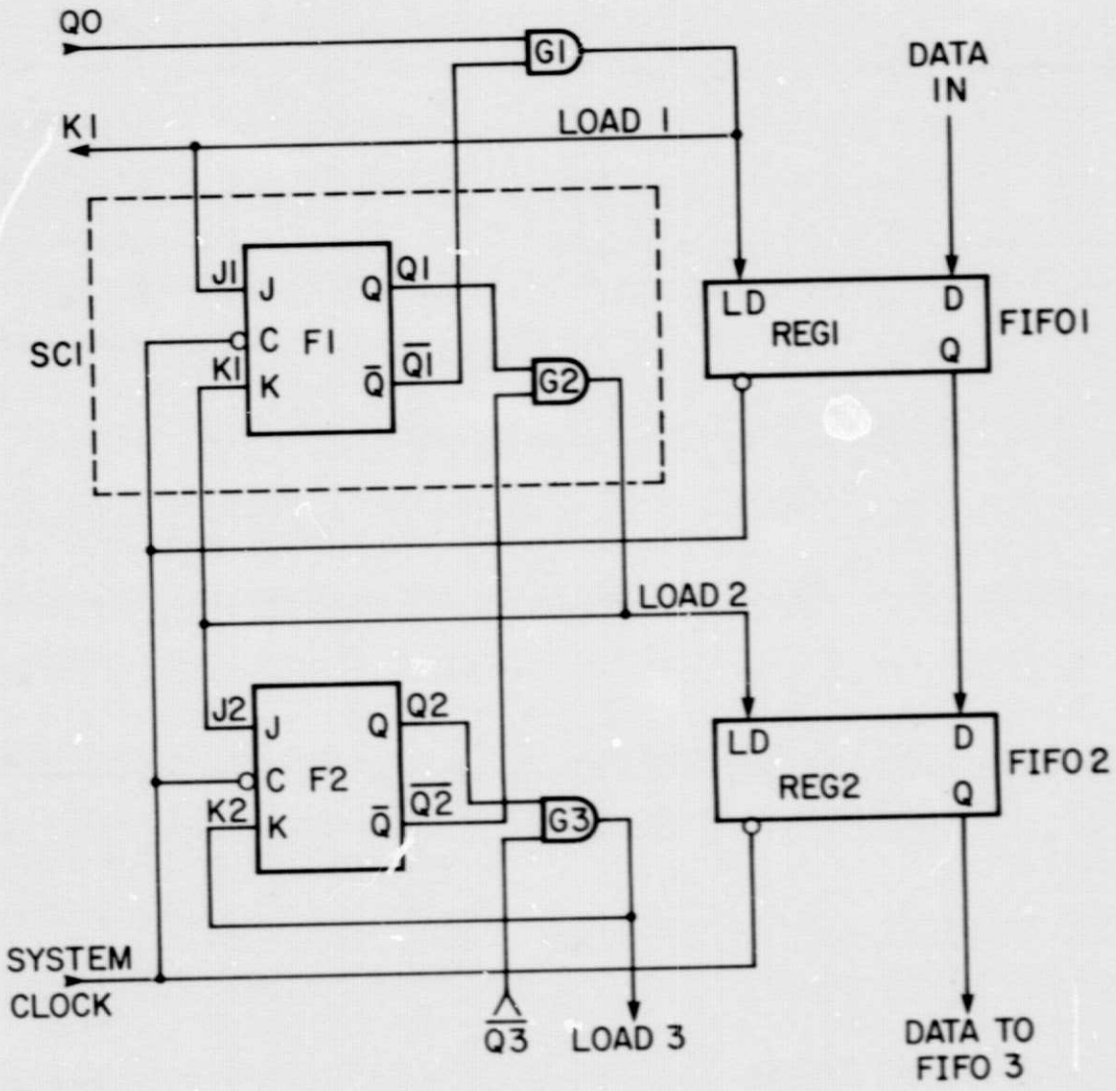

Figure 3.– Detailed logic showing two locations of the synchronous FIFO.

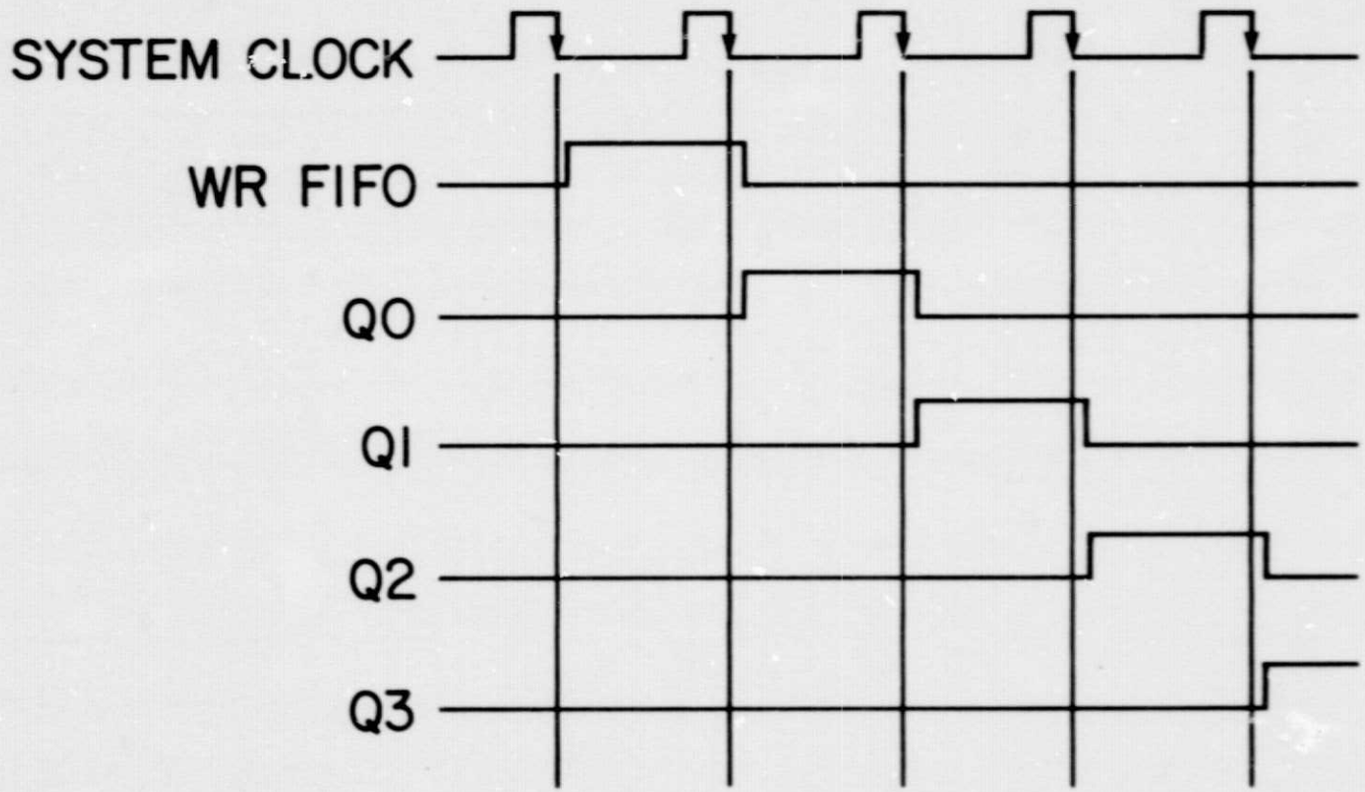

Figure 4.– Timing diagram for data written into synchronous FIFO.

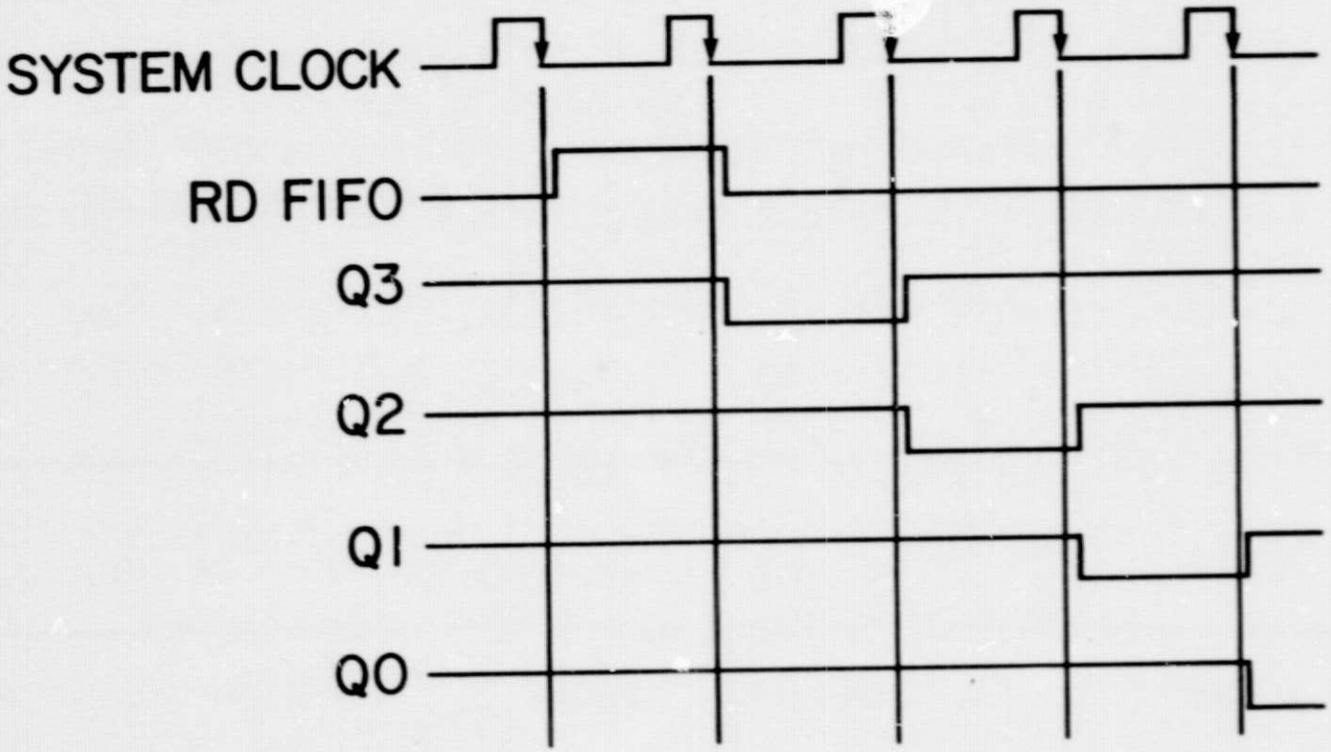

Figure 5.– Timing diagram for data read from synchronous FIFO.

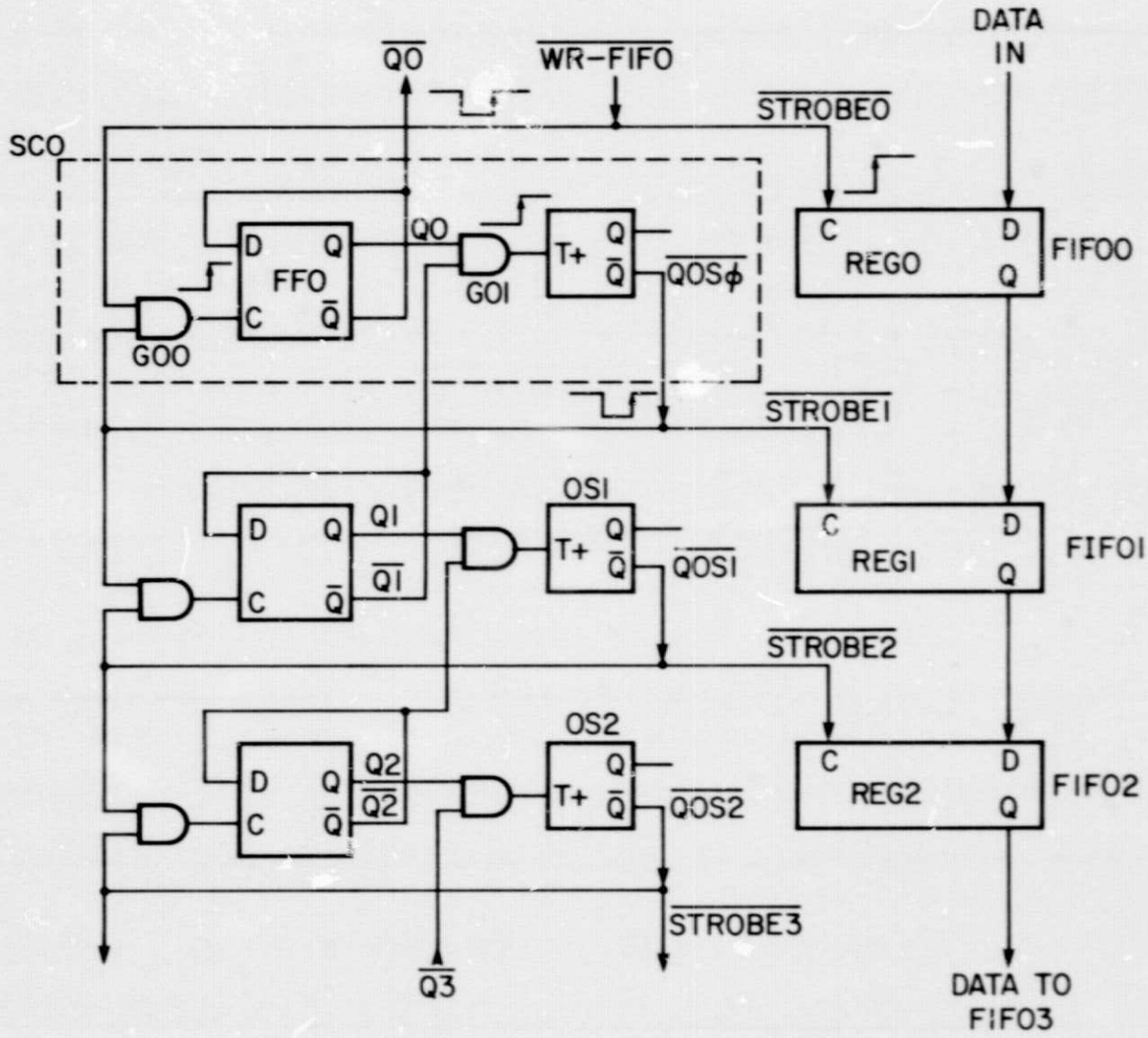

Figure 6.– Detailed logic showing three locations of the asynchronous FIFO.

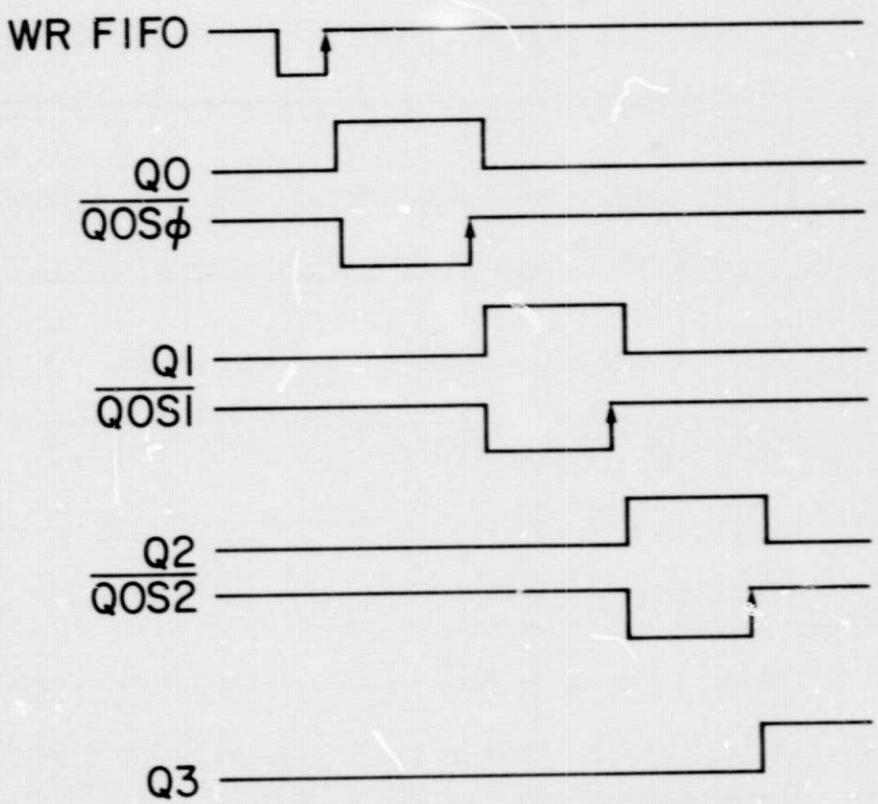

Figure 7.– Timing diagram for data written into asynchronous FIFO.

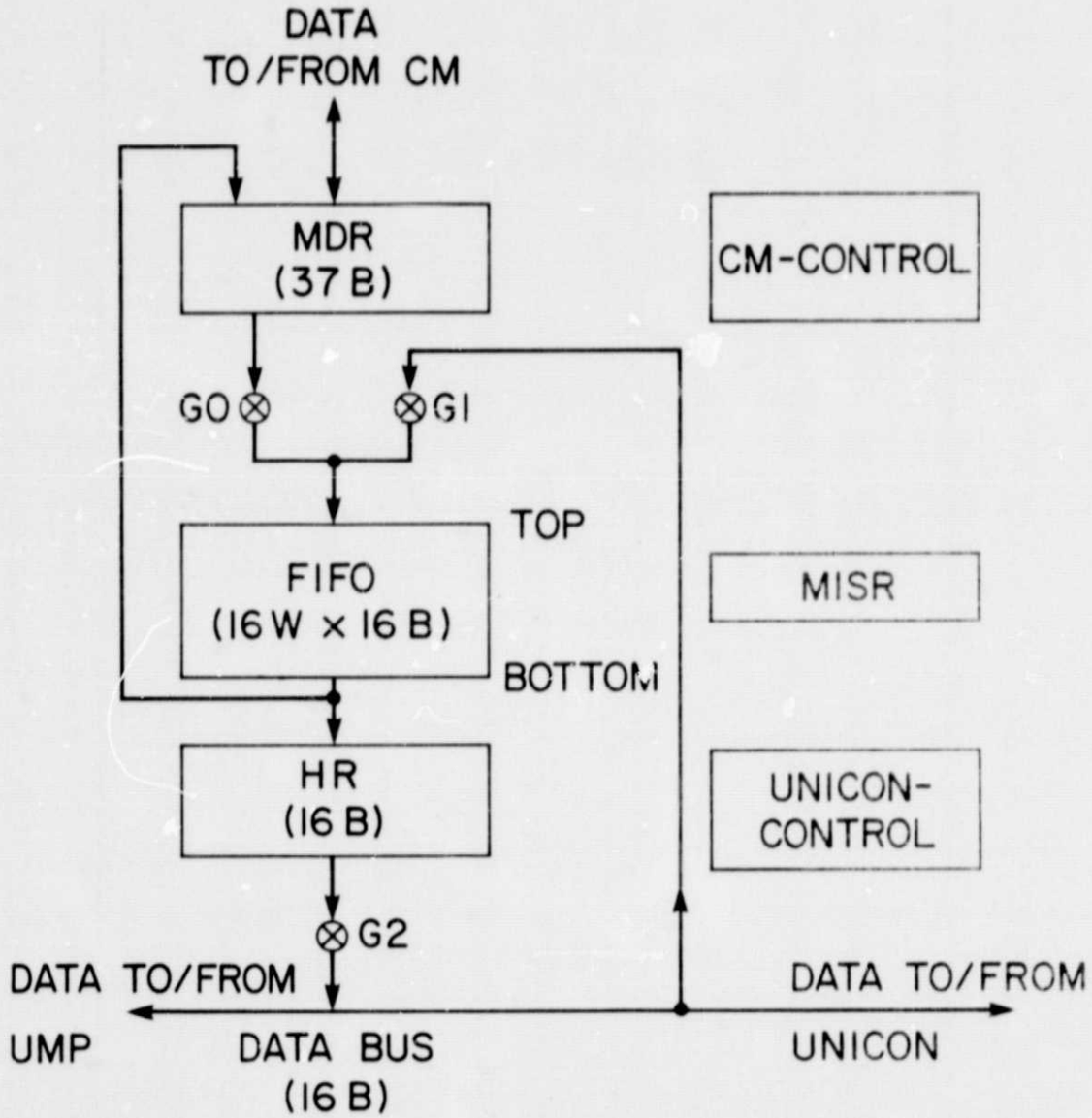

Figure 8.– UNICON Controller using a FIFO as a basic building block in its design.