NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

NL-339 (3/77)

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

MEMORY SYSTEMS FOR HIGH SPEED DATA PROCESSING

by

Z. H. Glanz

Submitted to the School of Graduate Studies
in partial fulfillment of the requirements for the degree
of
Master of Applied Science

Department of Electrical Engineering
Faculty of Science and Engineering
University of Ottawa
Ottawa, Ontario
June 1975

MEMORY SYSTEMS FOR HIGH SPEED DATA PROCESSING

ABSTRACT

A requirement for a memory for a modern high speed data processor is that it can be both direct-addressed and content addressed. The access time for content addressed data can be significantly improved if it is filed according to an appropriate set of descriptors. A further improvement can be obtained if hardware rather than software, techniques are used to implement the file, of sorting memory. The thesis deals with the design of a hardware implemented memory system which can sort and resort digital data, according to any set of descriptors chosen, in addition to accessing data on an associative or direct basis. The main limitation to the speed of sorting and resorting is the intercommunication system between the various parts of the store. The application of segmented ring bus techniques gives a high sorting speed and a flexible system. The memory design is suitable for LSI because it consists of a number of similar units capable of integration using silicon technology.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENT

The author wishes to express his thanks to his supervisor Professor P.M. Thompson for his guidance in carrying out this research work and the constructive suggestions he made in the preperation of this thesis. He would like to acknowledge Dr M. Krieger's contribution of many good ideas during useful discussions. Thanks go to P. Charlebois for his work on the realization of memory modules and to Microsystems International Ltd and General Instruments of Canada Ltd for providing the integrated memory circuits used.

CHAPTER I

INTRODUCTION

The speed of a digital data processing system can be measured as either the speed with which it performs some given set of operations or the speed with which data can be input. The first of these speeds is called the processing rate and the second the input rate or the flow rate. In some processors, such as those used in a control operation, the processing rate or the processing time is important, while in others, such as found in data communication, flow rate is important, because this is directly related to the band-width of the communication system. In this case, processing time is equivalent to a system delay and is similar to a transmission delay.

There are several ways of organizing a high-speed data processing system, two of which are given below:

a. Parallel organization: A "parallel" connection of arithmetic units, which usually use largely asynchronous logic.

b. Pipe-line organization: An organization in which data can be processed a single step at a time and the result stored, leaving the circuit free to process the next bit.

These two operations are not mutually exclusive and it is possible to use a parallel organization of pipeline units.

This thesis is concerned with the memory requirement of the above kinds of processors. A parallel organized system requires

rapid access to stored data, programs and sub-routines. This can be met by a conventional fast access store, providing it is supported by a high capacity, well organized paging system. Direct addressibility and content addressibility are both required. In real time high speed control, addressibility requires an appropriate sorting or filing system in the memory organization. A pipeline organized system has similar requirements to the above and, in addition, requires that data be presented at high speed and in the correct sequence.

Thus the memory characteristics that will be sought in this thesis are high information flow rate, direct addressibility, content addressibility, sorting and large capacity. The systems chosen should also be able to make use of the more modern technologies. To this end a modular organization is favored. Also serial memory sub-systems are preferred, because they appear to be the most cost effective per bit [1].

The thesis begins with a survey of modern memory technologies, followed by a short paragraph on content addressed memories and a review of data sorting methods. A hardware implemented sorting memory is then proposed, designed and its feasibility demonstrated. Finally it is shown that this can be developed into a general purpose memory system suitable for direct address, associative address and sorting.

CHAPTER II

MODERN MEMORY TECHNOLOGIES

The past decade is signified by the growth of electronic data processing which creates the need for larger, faster and more flexible memories. In the last two decades, on line storage capacity has increased about three times as much as C. P. U. power. Yet, memory represents the most limiting area in the development of more advanced computer systems.

Generally, memory devices can be classified into two basic categories: electronically accessed main memory, which is fast and relatively expensive, and electromechanically accessed peripheral memory, which is very slow and relatively inexpensive. Between these two widely different device technologies there is the "access gap". In recent years we find tremendous effort concentrated on magnetic bubbles, C. C. D. (charge coupled devices), electron beam addressed M. O. S. (BEAMOS) memories and holographic storage. The following is a review of recent activity in the development of the above technologies.

BEAMOS

A beam addressed metal oxide semiconductor is implemented to auxiliary memories, utilizing an electron beam to read, write and erase data on a simple MOS chip.

Information is stored on a memory plane consisting of several chips. The structure of a chip is shown in Fig. 1. It consists of a film of aluminum evaporated on a thin insulating layer, such as silicon dioxide, formed on a silicon diode. The n and p

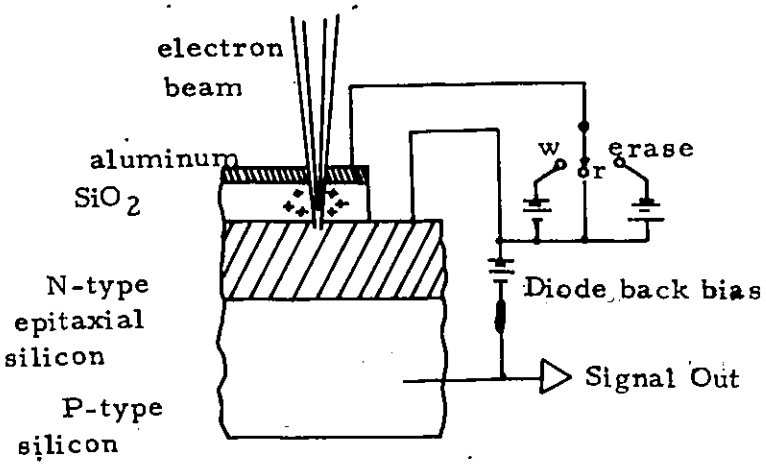
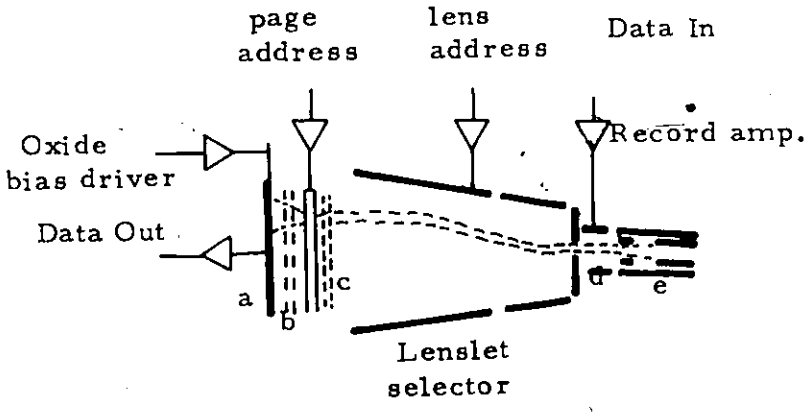


Fig. 1 Cross section of MOS memory chip.



a=memory plane; b=page selector; c=matrix selector;
d=beam modulator; e=electron beam source.

Fig. 2 BEAMOS electron optical system.

type silicon layers are connected electrically to form a back-biased diode. To operate, a voltage is applied across the oxide layer. In a typical system, positive 40 volts is applied to write and a negative 40 volts to erase. To read, 0 volts is applied.

To address an electron beam to a given location in a field, it is transferred through a single stage of deflection. Stability of power supplies and the accuracy of the deflection structure determine the number of addressable locations. A practical limit is a million addressable spots in a square field. In the BEAMOS module, addressing is achieved in a special electron optical structure.[8]

An electron optical system of BEAMOS is shown in Fig. 2. The first digits of the address are applied to a digital-to-analog converter which generates an analog voltage applied to the first deflection stage: lenslet selector. This directs the electron beam into one of the small lenslets, which is simply a pair of aligned holes in two metal plates. When applying voltage between plates an electron lens is produced, focusing the beam to an extremely fine spot. In addition, each lenslet has an individual deflection structure. The second part of the address is applied to another D-A converter; here deflection voltage is applied to a second deflection structure called: page selector.

To record data, lens and page addresses are entered simultaneously. This moves the beam to the beginning of the page. Now the beam is stepped across the lenslet field one bit site at a time while the electron beam is turned on or off as required by the input data.

Other operations may be performed by switching to the desired oxide bias and performing the same sequence. One BEAMOS implemented in a memory system features a 32 million bit capacity in one memory module. When more then one module is needed the electronics can be shared and hence the cost per bit is reduced. The access time is in the order of few micro-seconds and depends on switching and the setling time of amplifiers controlling the electron beam. In up-to-date modules, the data rate of recording is 10 Mbits/sec.

C. C. D. (Charge Coupled Devices)

The basic shift register nature of C. C. D. implies that its greatest application is in the high density semiconductor memory.

In C. C. D., the basic charge coupling principle consists of storing carriers in the inversion regions or potential wells under depletion-based electrodes, and of moving these carriers from beneath one electrode to beneath the next by appropriate pulsing of the electrode potentials.

To do this charge-transfer operation, the neighboring electrodes must be close enough to allow the potential wells between them to couple and the charges to move smoothly from one well to the next.

The inherent digital C. C. D. configuration is that of a shift register with a serial in-serial out operation, where all bits are simultaneously shifted. A C. C. D. memory element can be formed by using these shift registers directly or combining the shift registers

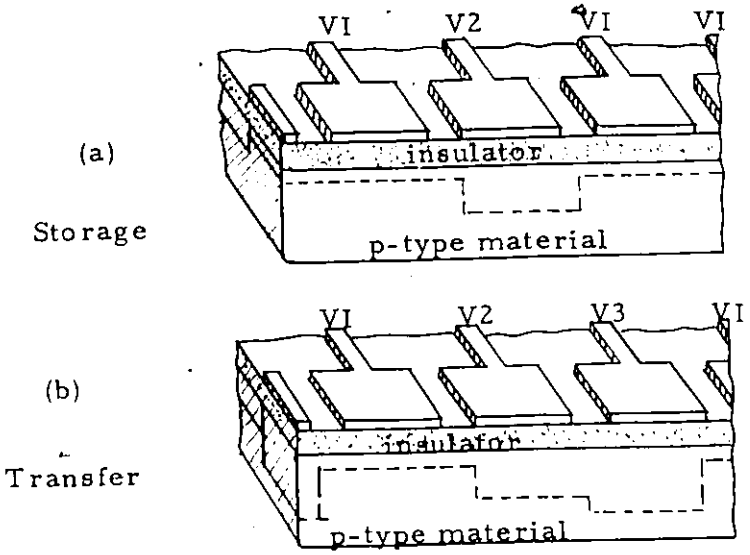


Fig. 3 Basic CCD. Charge is stored in a potential well, formed by voltage V2 larger than VI as shown in (a). Charge transfer (b) is accomplished by applying a voltage V3 greater than V2 , thus causing the charge to spill over.

with refresh, sense and decoding circuitry to form more sophisticated memories.

The C. C. D. storage element is dynamic and therefore must be periodically refreshed similarly to a dynamic MOS RAM.

There are three basic organizations for memory application:

1. Synchronous

Here all shift registers are clocked simultaneously. This organization (Fig. 4) is obtained by connecting several multi-stage shift registers by means of refresh cells internally. All data bits are simultaneously shifted through all cells of the register. The internal shift frequency is equal to the I / O data rate. Several such shift registers can be combined on a single chip using common control.

2. Serial-Parallel-Serial

An SPS memory arrangement is shown schematically in Fig. 5. The input rate depends on the clock frequency of horizontal registers. When the top horizontal registers are filled with data, a parallel operation is performed and all information is shifted to vertical registers.

3. Line Addressable

This is an integration of C. C. D. and MOS memory concepts. One form of such organization is shown on Fig. 6. Basically, the

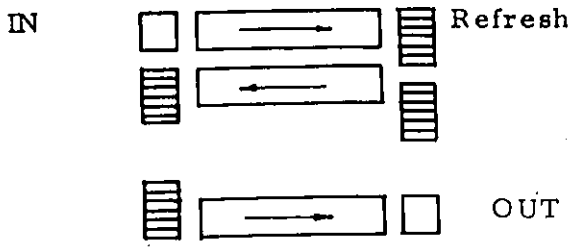


Fig. 4 Shift register organization.

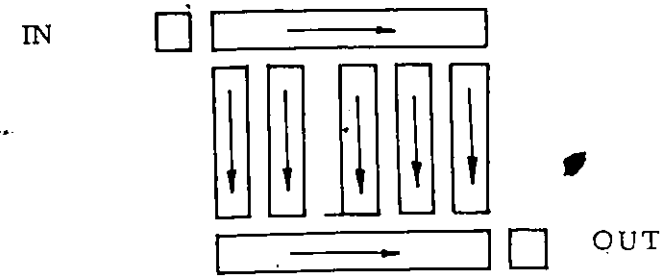


Fig. 5 Serial-parallel-serial shift-register organization.

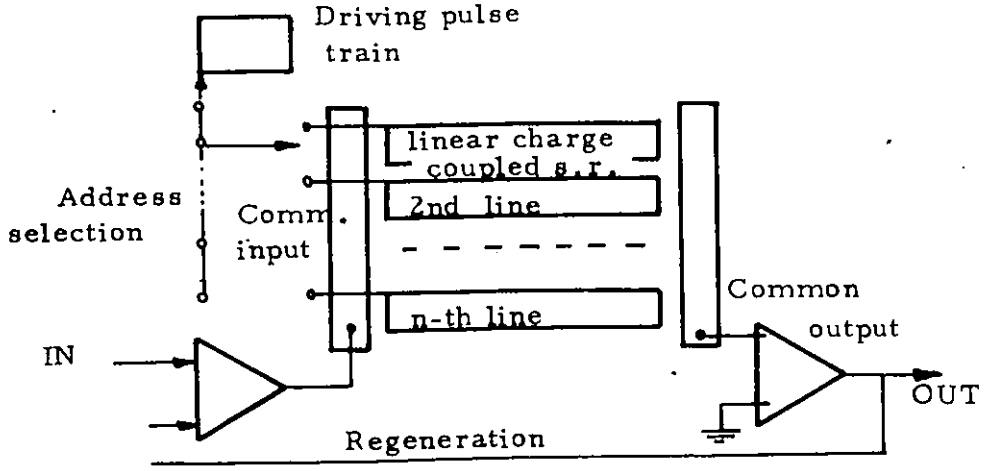


Fig. 6 Random acces line addressable memory organization.

memory is composed of an MOS address selection matrix and several C. C. D. shift registers each representing a line. Selection of an address causes the application of a driving waveform to a chosen line to initiate one of the common operations.

Holographic Memories

The holographic approach does not record individual bits but rather the optical interference pattern procuced by two coherent light beams, one of which contains the information about the data to be recorded. This is shown in Fig. 7. In general, the holographic approach requires that several unique components be arranged to produce a memory device.

These are a source of coherent light such as a laser, some rather conventional but sophisticated optical elements, a page composer to tranduce electrical data into a form which can spatially modulate a light beam, a recording medium which records the information as a hologram, an array of detectors to transduce the reconstructed light pattern back into electrical signals, and light deflectors or moving recording media to address various hologram positions on the recording media. The principal advantages of recording data in holographic form include:

(1) a natural distributive encoding by recording the information over the entire hologram rather than at discrete points, thereby reducing susceptibility to dust, scratches and recording media imperfections,

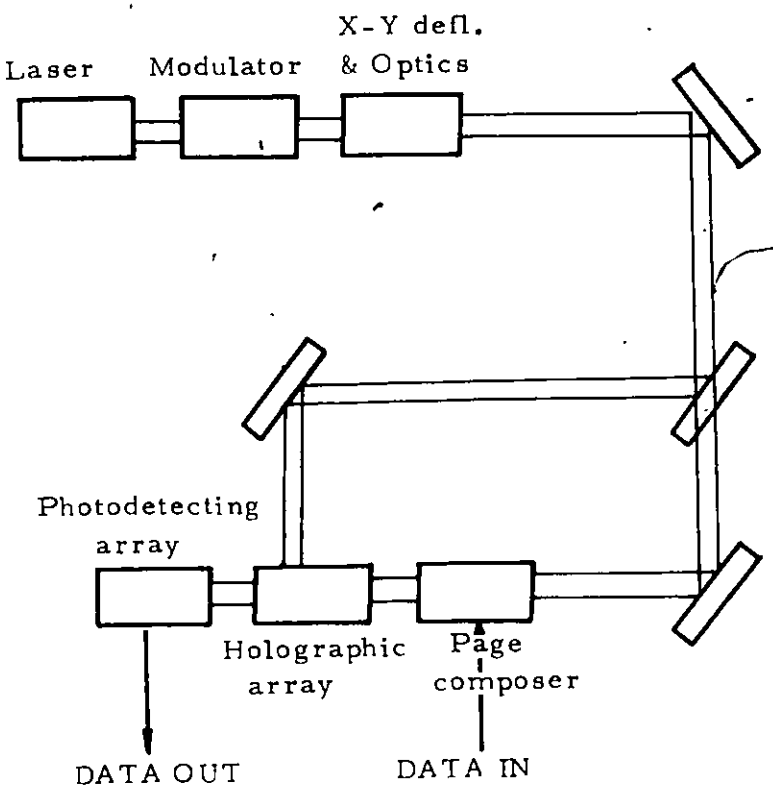


Fig. 7 Major electro-optic components in read/write holographic memory.

(2) the data reconstructed during readouts is projected directly onto a fixed photo detector array with 10^3 to 10^5 bits appearing simultaneously (i.e. a page-oriented parallel access), and

(3) insensitivity of the recording medium placement relative to the detector array.

A logical distinction between various holographic memories can be made considering those which are truly read/write memories and are aimed at existing mainframe, and those which are directed at main storage applications.

The major effort to date on read/write holographic memories has achieved capacities between 10^6 to 10^9 bits with access time measured in microseconds. Although progress on each holographic memory component has been significant over the past decade a truly viable cost competitive memory has not emerged. Because of the interactive nature of all holographic memory components, a major advance in one area may produce only minor improvement in system performance.

Read only holographic memories typically use film as the recording media. Once exposed, the film record is removed from the recorder, developed by normal techniques and placed in a holding area until data retrieval is required. Other than the recording media, the holographic exposive and data readout process are similar to those used in read/write memory.

Bubble Domain Memory

Since its debut in the late sixties, magnetic bubble technology has evolved to become a promising alternative to semiconductor technology.

To create a bubble domain, a crystal that serves as a source and host for the bubbles must first be grown. A rare earth such as thylium or terbium is placed in a crucible and a ferrite is added. Following a heating and cooling treatment, a single crystal magnetic oxide called orthoferrite is formed. It is used in the form of a platelet several mils thick. Orthoferrites can be magnetized easily in the axis normal to the plane (Fig. 8) but they are hard to magnetize in the plane. Under appropriate magnetic and dimensional conditions, cylindrical domains-magnetic bubbles can be formed and they remain stable over a range of bias-field values.

Magnetic domains respond to two opposing forces. One force, the magnetostatic energy, tends to increase the domain's surface area, while the other force, the sum of the bias field energy and the domain wall energy, tends to reduce the domain's volume and wall area. The domain wall takes a shape that minimizes the net energy, so the domain is cylindrical.

The bubble retains this shape over a range of bias values. When the bias increases beyond that which defines minimum diameter, the bubble is destroyed. As the bias is lowered the cylinder's diameter increases, causing the cylinder to take on an elliptical shape.

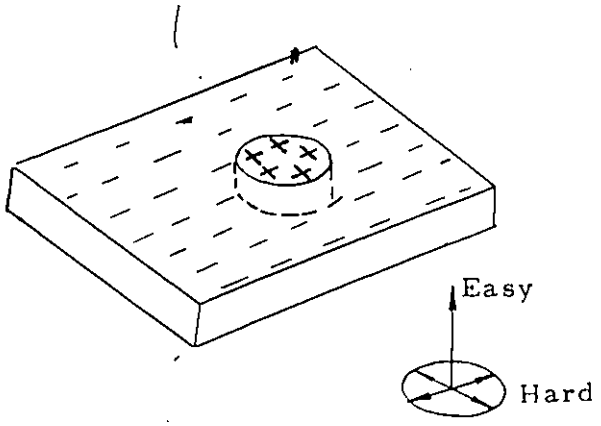


Fig. 8 Single crystal magnetic oxide

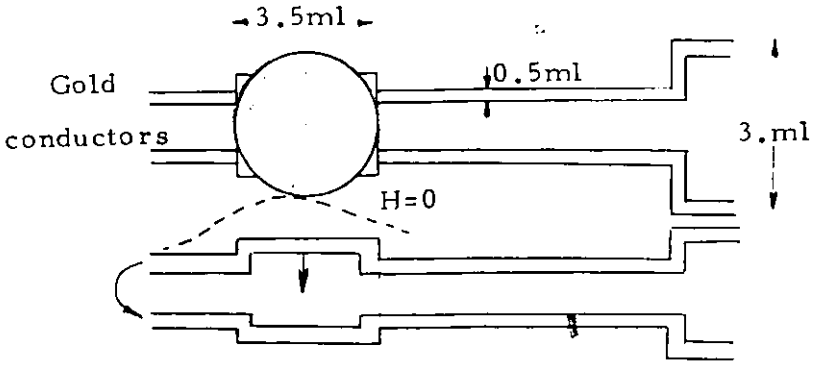


Fig. 9 Conductor drive

Current through lower square loop reduces bias relative to loop above, so magnetic bubble (dark disk) moves readily into new location.

Making the bubble smaller and larger through modulation of the bias field is a key mechanism in propagating a magnetic bubble across the plane of the platelet (Fig. 9). The presence or absence of a bubble in a particular location in the plane is the equivalent of either a 0 or 1 state.

Bubble domains move across the platelet in the direction of reduced bias. A cylindrical domain can move the distance of one domain diameter in less than 100 n seconds, so that the equivalent 10 mega-bit-per second clock rate permits a 3 mega-bit-per second data rate in practical devices. A practical application is a bubble (bit) source for a shift register. If the magnetic bubble shift register is large enough, it becomes a solid state version of a disk file. Data words can be inserted by generating and shifting a bubble for a binary 1, and not generating but shifting for a bubble train, then raising the local field by running a high current through the proper loop to collapse a bubble and obtain a 0. There are several characteristics which make bubbles attractive for storage systems: the bubble is non-volatile, providing protection for data during power failures, during operation, power consumption is low, bubble memory bit density is in the range of 2.5×10^6 bits per square inch.

A current implementation of bubble domain memory is FAM (fast auxiliary memory).

Magnetic Surface Memory - DISK

Magnetic surface memories are used extensively as backup storage in computers.

A magnetic head with a narrow gap between the poles is used for writing and reading of bits in much the same way as in tape recorders. A dual gap is sometimes used for immediate readout after writing, to check the recorded information.

The bit values are indicated by the direction of saturation magnetization and writing of these values is achieved by positive and negative magnetization currents through the magnetic head.

A disk coated with magnetic material is an example of a backing storage.

There may be as many as 50 disks driven together on one axis, which make 100 magnetic surfaces in one device. The tracks consist of concentric words, typically 100 of them on one side. The disks are usually removable. Some disk memories have only one pair of magnetic heads, one for all top surfaces and another for the bottom sides (Fig. 10). Sometimes there is one arm for each disk. To search a word from a disk, three steps are necessary: searching the surface, moving the arm to a desired track, and waiting for the desired sector. Words can be arranged in series and parallel. On disks the series method is more commonly used; the consecutive bits of one word are recorded serially on one track. To specify a word it is thus necessary to indicate the track and the relative position of the word on it, measured from a reference point.

The most recent development in small disks is the flexible or floppy disk file (Flexydisk).

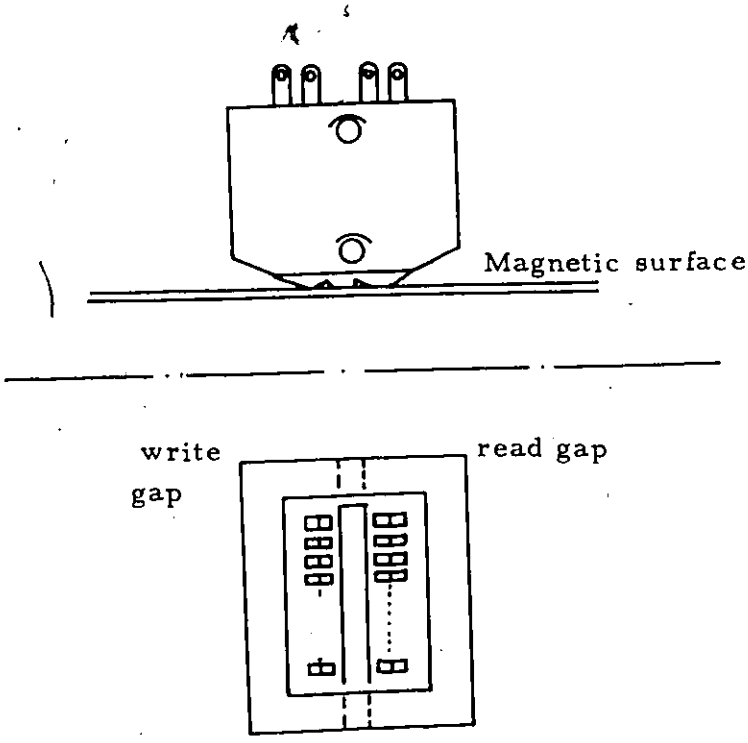


Fig. 10 Dual gap magnetic head(for nine tracks)

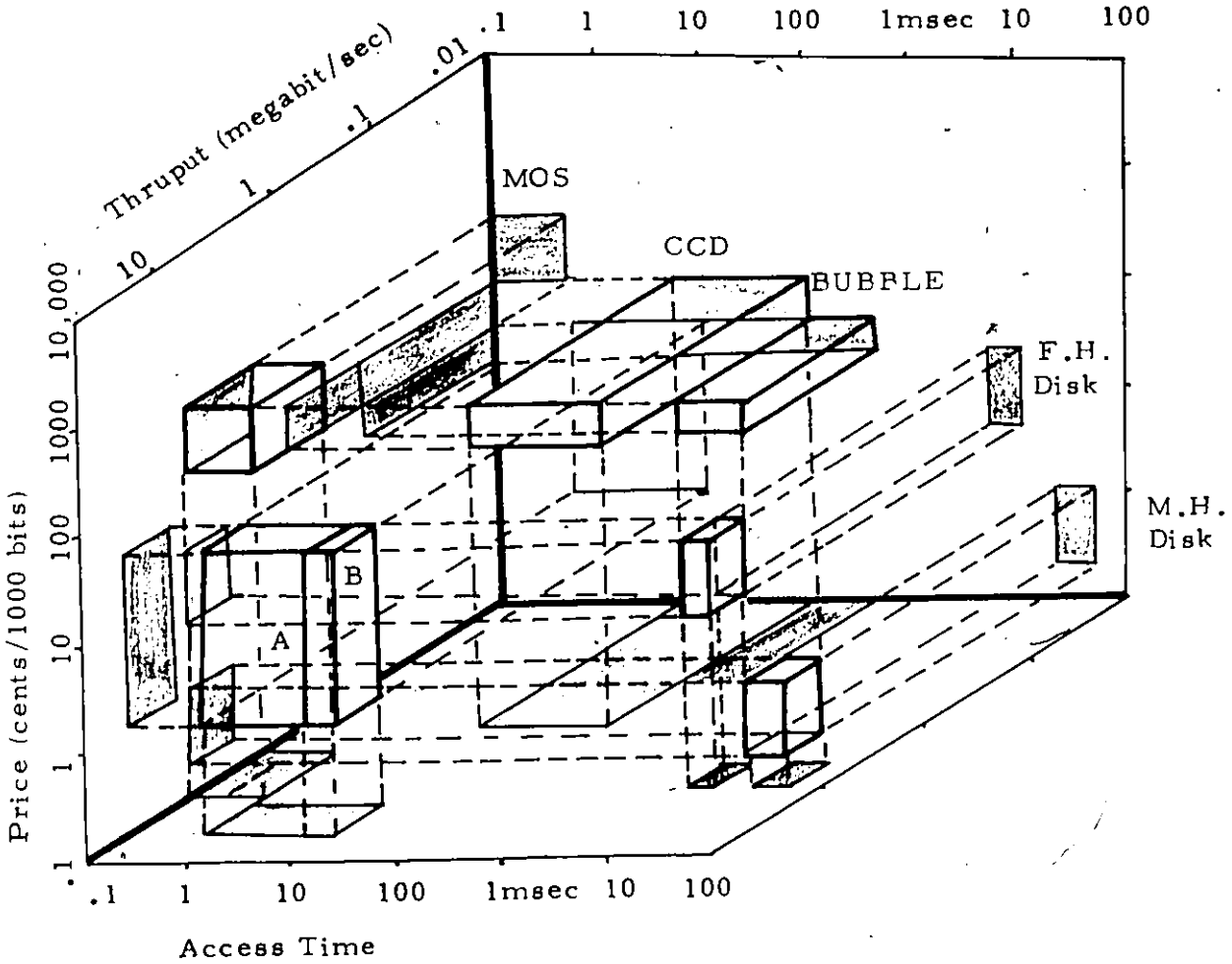
A flexible disk is a cartridge consisting of a flexible magnetic disk enclosed in a semi-stiff plastic jacket. The disk is free to rotate within the jacket for reading and writing via an in-contact head which requires that the media also have all of the virtues of a superior computer tape.

The advantages of floppy disks are in fast random access and high-speed transfer at low cost for megabit or higher capacity. Rotational speeds of up to 60 r/s have been announced. Total random access ranges from 100 msec to over a second. Data rates range from 30k bits/sec to 2.5M bits/sec. Capacity is usually 1M to 2M bits. Recently a flexy disk product was announced having a capacity of 252,928 bytes on one recording surface having 77 tracks. The total thickness of the disk was .07 inches.

The above is not a review of all the technologies but of the novel ones.

Ferrite-core memories will be with us for some time yet, if for no other reason than sheer momentum. Other technologies, as for example MOS memory, are already well established in many applications.

Memory performance and cost are the twin keys to computer technology. Fig. 11 shows price-performance comparisons for several memory technologies to the end of this decade. Memories as projected become larger and faster to cope with the requirements of computer systems.



A- Micro-bit+ MOS CACHE
B- MOS CACHE

Fig. 11 Technology comparison through 1980

CHAPTER III

CONTENT ADDRESSED MEMORY (CAM)

The concept of CAM goes back several years. A content addressable, or associative memory is one from which data is addressed in terms of its content rather than its location. Alone among memory forms, the associative requires both storage and logic in its arrays. CAM is associated with the connection of match lines. Each match line must be individually connected to all cells in its row. To use CAM, each word entered into the memory contains a "descriptor" as well as a data portion. In some systems what is considered a data portion at one time may at a later time be used for searching.

Some organizations may permit multiple matches when searching a CAM. For example, the search may be equivalent to the question, "How many entries do I have which used X as the descriptor?". And the result would be a selection of several data words, if several such entries were found.

Associative memories have been designed from several memory technologies. A few of them are summerized here.

Magnetic Techniques

Bubble memory is one magnetic technology that has not been tried for CAM's but may prove valuable for large associative memories.

Bell Labs and IBM have made some predictions of the characteristics of bubble memories in this application:

- up to 20, 510 bits per chip
- 100K c/s data rate
- non volatile
- .01 to .001 ¢ per bit

Bubbles are serial devices but access speed can be greatly increased by on chip multiplexing techniques.

Associative Memories Using Holography

Holography is well suited to associative memories because of its inherent exclusive-OR facility. Two coherent beams of light are used to illuminate a photographic plate. At a later time a third beam of light illuminates the plate, and the diffracted light has characteristics of all three beams. In particular if one of the two original sources is used as the "read" light, the other original source can be easily derived from the diffraction pattern. For example, the beam could be encoded to represent a man's name and his salary. At a later time the salary beam could "read" the plate, and the name would be diffracted.

The major disadvantage of holography is the difficulty in storing information. As such it can really only be considered for read-only associative memories.

Semiconductor Associative Memories

In recent years semiconductor devices have taken the lead in associative memories. Types of technology include TTL, MOS, Schotty TTL and CMOS.

The general advantages of semiconductor technologies are:

(1) High signal to noise ratio

(2) Low power requirements

(3) Compatibility with external logic

(4) Capability in logic-in-memory design.

The general characteristics of the various technologies are:

Technology	Access Time	Density	Power
Bipolar TTL	50msec	19.9 mil^2/gate	20mW/gate
Schotty TTL	20		60m "
CMOS	500	49.8	1μ "
NMOS Silicon	200	5.6	100μ "
PMOS	500	10.6	100μ "
Bipolar I^2L	50	4.8	100μ "

A typical high-speed content addressable memory is organized as 16 eight-bit words. The entire device is constructed on a single monolithic chip using low threshold MOS P-channel enhancement mode transistors.

Active-element design permits non-destructive read out and interrogation of memory contents. Bit lines can be wire-OR connected to obtain memory planes greater than 16 words.

Word lines can be wire-OR connected to achieve word length greater than eight bits per word. Selection of a given word for reading or writing is accomplished by connecting the selected word line to a negative voltage while holding all other word lines at ground. The common interrogation control 1, when returning to a negative voltage, allows all sixteen words to be interrogated simultaneously. Writing is accomplished by addressing a desired word and bringing the appropriate bit lines to ground while holding the other bit lines at a negative potential. If both bit lines of a selected bit are held at the negative potential, the information in the bit will be unchanged. Masked writing can thus be achieved. Reading each bit content of an addressed word requires sensing a different current between the two bit lines. (Fig. 12)

Interrogation of memory content is accomplished by activating the interrogation command 1, bringing bit lines to appropriate voltages, and simultaneously sensing the current in each line. If both bit lines of a particular bit are held at ground potential, that bit will be excluded from interrogation. If a word matches perfectly, no current will flow through the word lines.

A moderately intensive search for available associative memory chips reveals only few such products on the market; these are presented in the Appendix A.

CONTENT ADDRESSABLE MEMORY ORGANIZATION

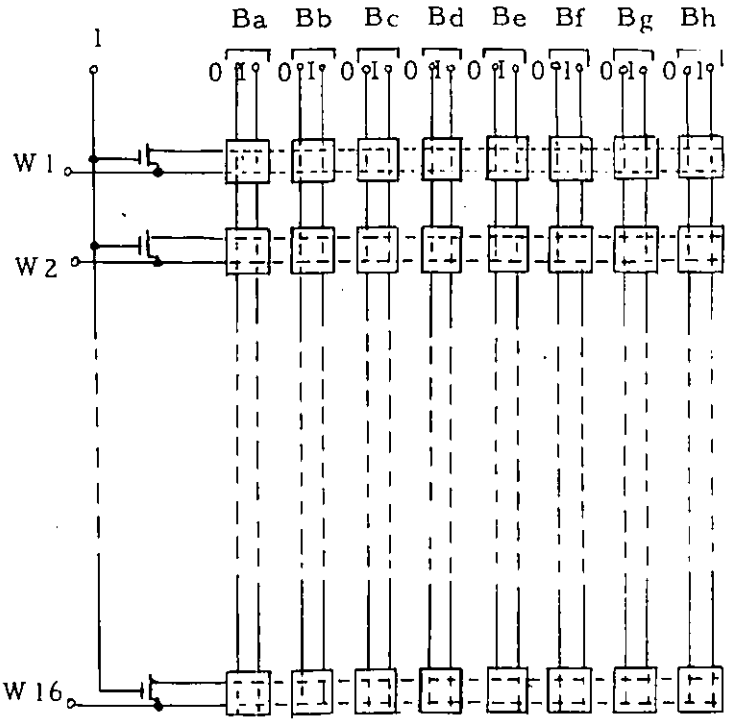


Fig. 12

CAMs find application in buffer memory control and implementation of some functions, such as sorts, searches, etc. Content addressable memory function can also be realized with much longer effective access times, by actually searching the memory. For example, a shift register memory may be designed to include the comparison function as part of the recirculation logic. The data is completely searched in one circulation of the memory. The low capacity of CAM, in the above and other applications, limits its implementation to handling small amounts of data.

CHAPTER IV

REVIEW OF DATA SORTING METHODS

The advent of large internal memories has increased the relative importance of sorting, because with large internal memories fewer applications require an external sort, and even if external sorting is required, the internal sorting of large segments can take a substantial amount of time.

It seems useful to survey the methods available for internal sorting in order to choose methods suitable for general use. Almost all methods of internal sorting use software.

The process of rearranging a file so that is is "in order" with respect to a descriptor mapping is called sorting the file. Different sorting algorithms can be compared from several different points of view: computer speed, program complexity, storage and peripheral resources required, etc. Some of these sorting methods require that all information to be sorted be accessible to the computer before the sorting is begun. Other methods process each item of a file individually in its original order so that later items need not be accessible to the computer.

Software Methods

Software methods for sorting may be distinguished by the amount of storage required. This storage must include the sorting program itself, and the allocation for the file to be sorted (one or more copies). Some methods require also additional storage for pointers to locations of file items.

Here we are interested in internal sorting in which the whole file is in the internal memory of the computer so that all items are equally accessible. The time T(n) for an internal sort of a file of n items depends on a number of factors, including:

(1) The method used
(2) The efficiency of the program
(3) The characteristic of computer
(4) The original order of the file
(5) The length and/or structural complexity of file items.

The following sorting procedures will be reviewed.

1. Upward/Downward radix sort
2. Bubble sort
3. Partitioning: Quicksort
4. Address sorting

We can then compare the reviewed methods, using time and storage requirements as the scales.

Upward Radix Sort

A method of demonstrating this sorting technique is to give the algorithm implemented in software:
Although the example given is for four digit word, there are similar algorithms for other word lengths.

1. Convert the items in the file to binary and separate into two lists according to the last digit
2. Separate again using the next to last digit
3. Repeat using the second digit
4. Repeat the same operation using the first digit
5. Convert back to decimal.

Example: (4 2 7 5 8 1 3 9 6)

1) (0100 0010 0111 1000 0001 0011 1001 0110)
(0100 00010 1000 0110) (0111 0001 0011 1001)

2) (0100 1000 0001 1001) (0010 0110 0111 0011)

3) (1000 0001 1001 0010 0011) (0100 0110 0111)

4) (0001 0010 0011 0100 0101 0110 0111) (1000 1001)

5) 1 2 3 4 5 6 7 8 9

Downward radix sort is a similar algorithm, starting with the first digit.

Bubble Sort

Here the file is scanned several times comparing adjacent items and exchanging if necessary, in this way the largest item is "bubbled" to the end of the segment being sorted.

Example: (4 2 7 5 3)

4 2 7 5 3
2 4 7 5 3
2 4 7 5 3
2 4 5 7 3
2 4 5 3 7
2 4 5 3 7
2 4 5 3 7
2 4 3 5 7
2 4 3 5 7
2 4 3 5 7
2 3 4 5 7

Partitioning: Quicksort

Given any unsorted segment of a file a bound element is chosen, forming a lower segment consisting of those items less than or equal to the bound and an upper segment with items greater than the bound. The bound itself is left between these two segments. If at any time, there is more than one unsorted segment left, the shortest one is partitioned

Example: Given the file (4 2 7 5 8 1 3 9 6) and the bound element in the centre (underlined).

(4 2 7 5 1 3 6) 8 (9)

(4 2 7 5 1 3 6) 8 9

(4 2 1 3) 5 (7 6) 8 9

(4 2 1 3) 5 (6) 7 8 9

(4 2 1 3) 5 6 7 8 9

(1 2 (4 3) 5 6 7 8 9

1 2 (4 3) 5 6 7 8 9

1 2 (3) 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9

The algorithm is easily implemented using push down lists.

Address Sorting

Address sorting depends upon the presence of an acceptable address calculation operator W. Each file item is put into a location determined from its descriptor by address calculation using W.

Example: Given the file of nine items

(4 2 7 5 8 1 3 9 6)

we might provide 15 such locations. Putting each item X in turn into the location [1.5 x X], we obtain the following sequence

				4										
	2			4										
	2			4				7						
	2			4	5			7						
	2			4	5			7		8				
1	2			4	5			7		8				
1	2	3		4	5			7		8				
1	2	3		4	5			7		8		9		
1	2	3		4	5		6	7		8		9		

If this sequence is copied back into the original places omitting blank spaces the sorting is completed.

The efficiency with which the address sorting works depends upon the relative size of working storage in main memory. If items are placed close together so that there are not more output locations than there are input items, the sorting becomes inefficient, which arises when we try to place an incoming item in a spot that is already occupied.

Case 1: "perfect aim"

The address calculated is empty. The incoming item is placed in the calculated position.

Case 2: occupied place; room at the top

The calculated address is full and the descriptor at that address is smaller or equal to the item descriptor to be placed. Hence the new item belongs to higher address, and its placed in the next address.

Case 3: "squeeze in" the item

The calculated address is full as are several succeeding addresses; more important, the desired item must be placed between two already filled locations. We find a vacant address and move the contents of each address after the one where the desired item belongs. Now it is possible to insert the item in its proper location.

The above methods for sorting are compared in the table below.

SORTING METHOD	PROGRAM BYTES	ADDITIONAL STORAGE	TIME SECONDS
u/D Radix	690 1270	40 k	4.43 4.4
Bubble	1450	20 k	3.5
Quicksort	1576	-	.801
Address	1034	60 k	.415

NOTE: These results were obtained using PL-1 program to sort 10,000 words.

Since the publication of the above sorting methods, more advanced software techniques have evolved [ref. 15 (a)] however, they don't overcome the essential slowness of the sorting approach.

Hardware Method

Another approach to obtain sorting capability is to combine logic and memory functions in basic building blocks. These blocks are arranged in regular arrays, commonly called Logic-in-Memory (LIM) arrays [16] . These arrays may be regarded either as a logically enhanced memory arrays or as a logic array whose cells can be programmed to realize a desired logical function. The main advantage of these arrays is their suitability for LSI, because of the highly modular structure of the arrays and because digital machines can be built using few types of these arrays.

Sorting Arrays

A sorting array is a multiple-word memory that keeps in sorted order all data words that are fed into it. Words that are read out are obtained in order of size, with the largest word first (or, alternatively, with smallest first, as desired). The words are assumed to be a maximum length n. This array can be used as a functional unit within the central processor of a general-purpose computer, or to a special purpose computing system for which sorting capability is needed.

The sorting array described here [16] is a two-dimensional iterative configuration of identical cells, each of which is a simple sequential logic circuit. Each cell is connected only to its immediate neighbors and the cells around the edges of the array are connected to fixed signals or to registers. Fig. 13 shows the array configuration along with an input-out-ut register X and a word selected register W. All cell input terminals on the right-hand side of the array are connected to a logical signal Zo, (normally fixed at 1) and the outputs labeled $\hat{Z}$ from the left edge of the array serve as inputs to the stages of the W register. Each cell (Fig. 13) contains one flip-flop, whose content is designated Y, so that the set of n flip-flops in any one of the N rows of the array may be employed to store one n-digit word. This set of words is assumed to be encoded. A binary-coded decimal number system can be used with the most significant digits at the left ends of the words. One's or two's complement representation can be used to encode negative numbers, with the minus sign encoded as a O at the left end of the word.

An entire word is handled as a unit during the input, output and sorting operations. One cycle of operation of the array consists of two steps:

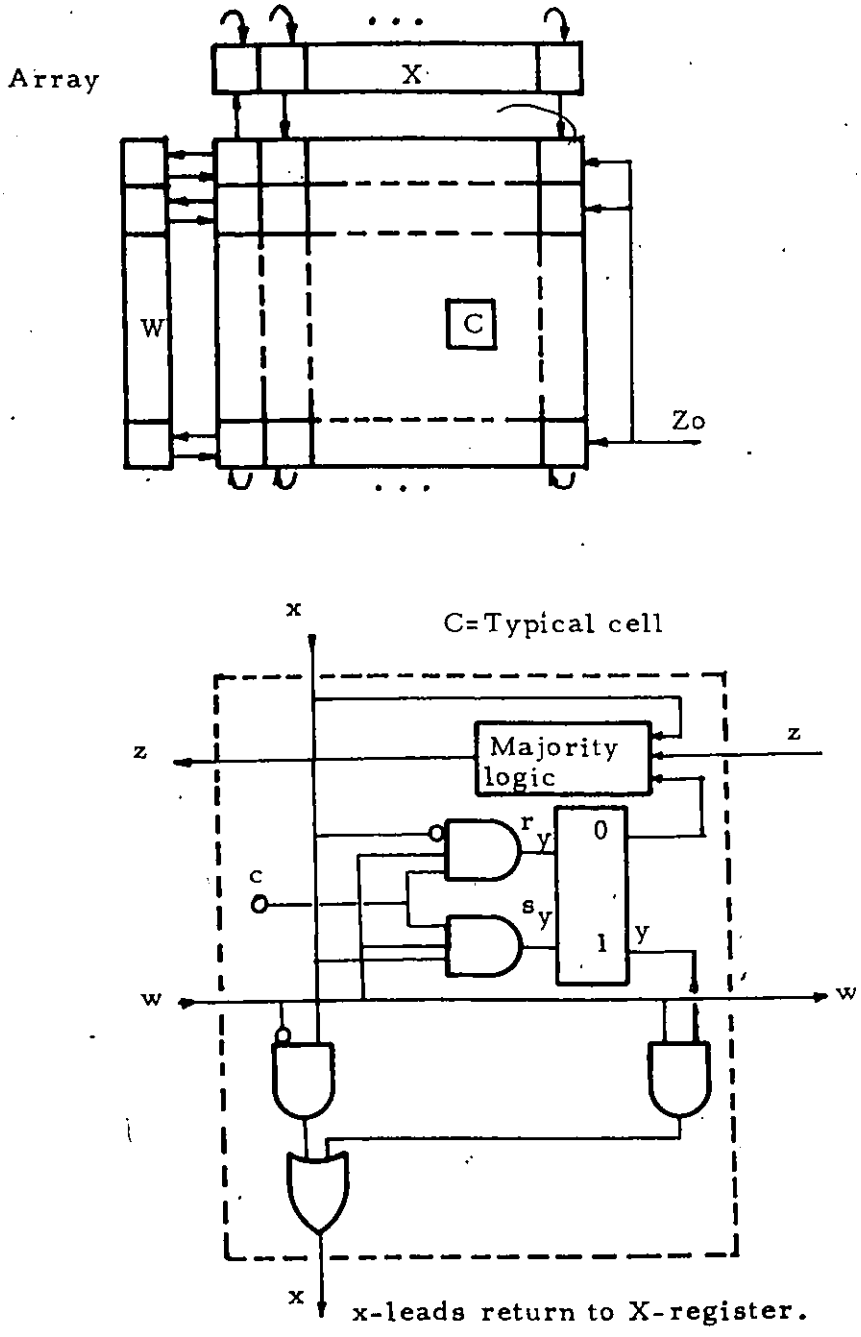


Fig. 13 Cellular Sorting Array.

(1) A comparison step, in which the word X in the X-register is simultaneously compared with N words stored in rows of the array; a " 1 " is injected into the W register in those rows whose words (including, blank words) are smaller than or equal to the word X, and a "O" is injected into the W register in those rows whose words are larger than the word X.

(2) An execution step, in which

(a) The set of all words that are stored in the subset of the rows having a " 1 " in the W-register are moved downward one row within the set, while the word in the X-register is copied into the uppermost such row; and

(b) The lowermost such word is copied into the X-register. Words having a "O" in the W-register are not moved.

Sorting with this array is accomplished by maintaining a sorted file of previously entered words, with the largest at the tope of the array and the smallest and any blank rows at the bottom. Each new word that is to be sorted is inserted into this file in a single operation cycle. To read out in order the words in the file, largest to smallest, place a " 1 " in the uppermost stage of the W-register. Now carry out step (2) of the cycle repeatedly, shifting the " 1 " downward in the W-register, one row with each step. If the words are desired in the opposite order, a " 1 " may be started at the bottom of the W-register and shifted upward.

The advantages of LIM array are:

(i) Fast bit-parallel search and sort

(ii) Single IC simplicity

(iii) Potential use of universal logic module.

The disadvantage is:

(i) Low bit capacity i.e. for systems processing large amounts of data LIM's implementation becomes cumbersome.

CHAPTER V

A DATA SORTING SYSTEM USING A HIGH SPEED BUS

The access time for content addressed data can be significantly improved if it is filed according to an appropriate set of descriptors. A further improvement can be obtained if hardware, rather than software, techniques are used to implement the file, or sorting memory.

A sorting array has some of the characteristics of an associative memory, but the position of an entry in the array depends upon a characteristic of the entry, e.g. several descriptor bits masked positionally. For i descriptor bits there is a maximum of 2^i sets of entries. A characteristic of a sorting array is that the entries are sorted into new sets, if the position of the mask for the descriptor bits is changed. For example, this thesis might be filed under one of several index terms and the choice of a new index term would result in a change of file. In many applications, the speed with which data entries can be resorted is important and software techniques prove to be slow.

It follows that there is a need for a hardware implemented sorting system. Koutz [16] designed a Logic-in-Memory system, described in Chapter IV, to deal with the sorting of small amounts of data. Associative memories can also be used within a sorting system, but there are difficulties associated with the communication between the seperate associative memory units. Indeed, it seems that the main limitation to the speed with which a resorting operation

can be performed is the flexibility and speed of the intercommunication system between the different storage units in the overall system.

Segmented Bus

A solution to the intercommunication problem can be found in the segmented bus [18]. For a sorting array the segmented bus is used in its simplest form, where it is similar to a parallel shift register one word wide with various inputs and outputs along its length (Fig. 14). Words are transmitted by being clocked from segment to segment along the bus in "carriers".

Words can be entered, at input/output segments or ports, if there is an empty carrier. It is possible to enter several words, at the same time, if there are empty carriers at the appropriate ports. Similarly, several words can be output at the same time. When a word is output, its place can be taken by a new input word. Thus the segmented bus provides communication between a pair of ports, without preventing communication between another pair, and it can provide communication between several pairs at the same time. It is usual to clock the shole system, bus and input/output units, at the same speed.

Simple Ring Configuration

A simple arrangement for a sorting memory is shown in Fig. 15 A segment of the bus is extended to become a memory segment, which consists of a memory unit and a "coupling Filter" associated with the appropriate segment of the bus. An additional segment is used as an input/output unit.

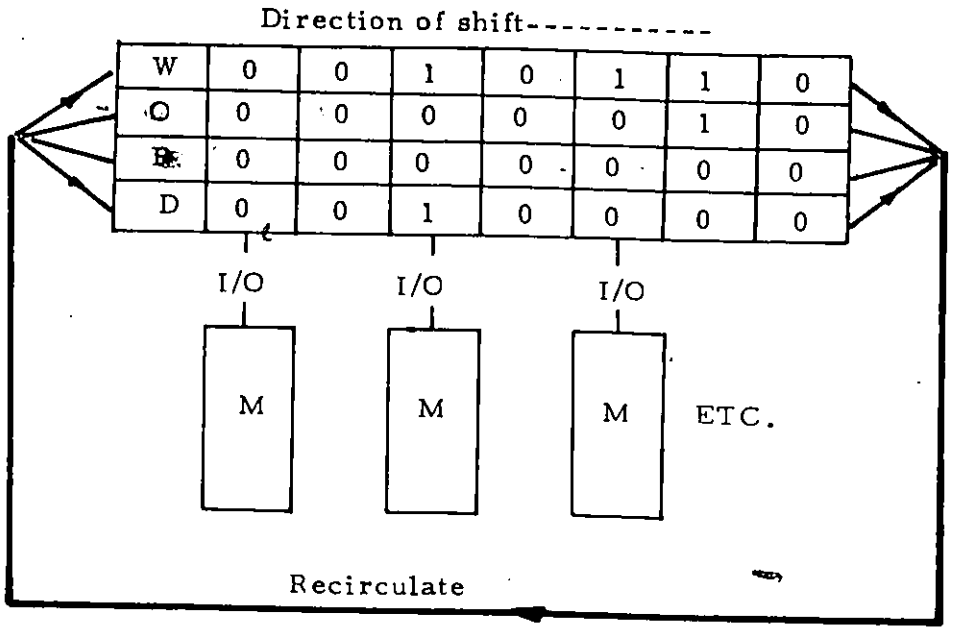


Fig. 14 Simple segmented bus (Shift register)

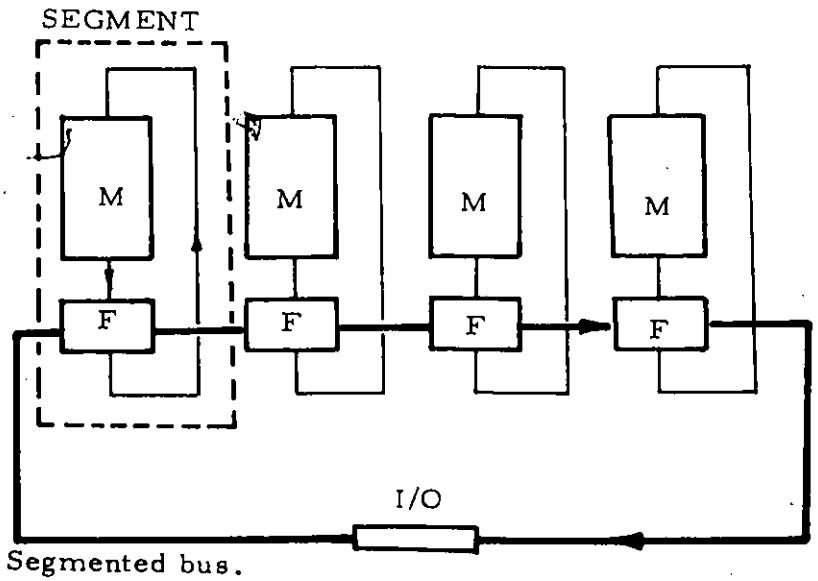


Fig. 15 Sorting Ring

This ring can perform three basic functions:

(a) Input information can be clocked around the ring until it reaches an appropriate memory segment, where it is stored.

(b) A resorting operation may be performed by changing the descriptors for the different memory units.

(c) Words may be output on an associative basis.

In Fig. 15, the memory units are first-in-first-out (FIFO) stacks and the coupling filter (Fig. 16) is routing logic, which can perform the following operations:

(a) Words arriving from the left are routed to the top of the stack, if they contain the appropriate descriptor for that stack.

(b) Words that do not contain the appropriate descriptor are routed straight through to the right.

(c) Whenever an empty carrier enters the segment, a word at the bottom of the stack can be routed to the right, if it does not contain the appropriate descriptor for the stack.

(d) Words at the bottom of the stack that contain the appropriate descriptor are routed straight through and returned to the top of the stack.

Complex Configurations

The number of memory units on a ring grows with the number of words to be stored and descriptors used. However, although the

COUPLING FILTER FUNCTION

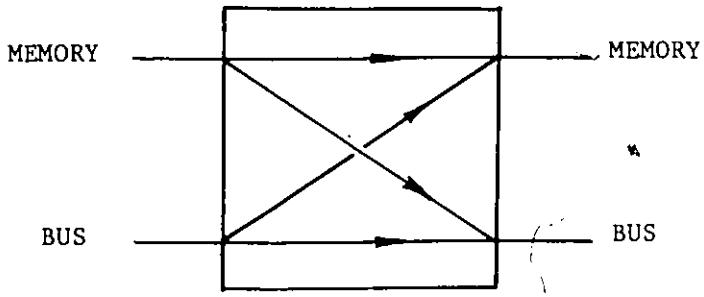


Fig. 16 TRANSFER BUS TO MEMORY STREAM IFF:

1) THE DESCRIPTOR IN THE MASK MATCHES.
2) THERE IS SPACE IN MEMORY STREAM.

OTHERWISE IT REMAINS ON THE BUS.

TRANSFER MEMORY TO BUS STREAM IFF:

1) THE DESCRIPTOR DOES NOT MATCH.
2) THERE IS SPACE IN BUS STREAM.

segmented character of the bus allows multiple entry and exit of data words, the single channel characteristic of the ring limits the information flow and results in extended sorting and resorting times.

The speed of sorting and resorting can be improved by complex bus configurations. A good criterion for comparison of these configurations is the maximum resorting time, because this is a well defined parameter and always larger than the initial sorting time. For a simple ring bus, the maximum resorting time is given by:

$$T_{(max)} = W_m \cdot M_n \cdot t_c$$

where $T_{(max)}$: maximum resorting time

W_m : No. of words in a memory unit

M_n : the total No. of memory units

t_c : clock period.

If a multiple ring configuration, such as that shown in Fig. 17, is employed, then:

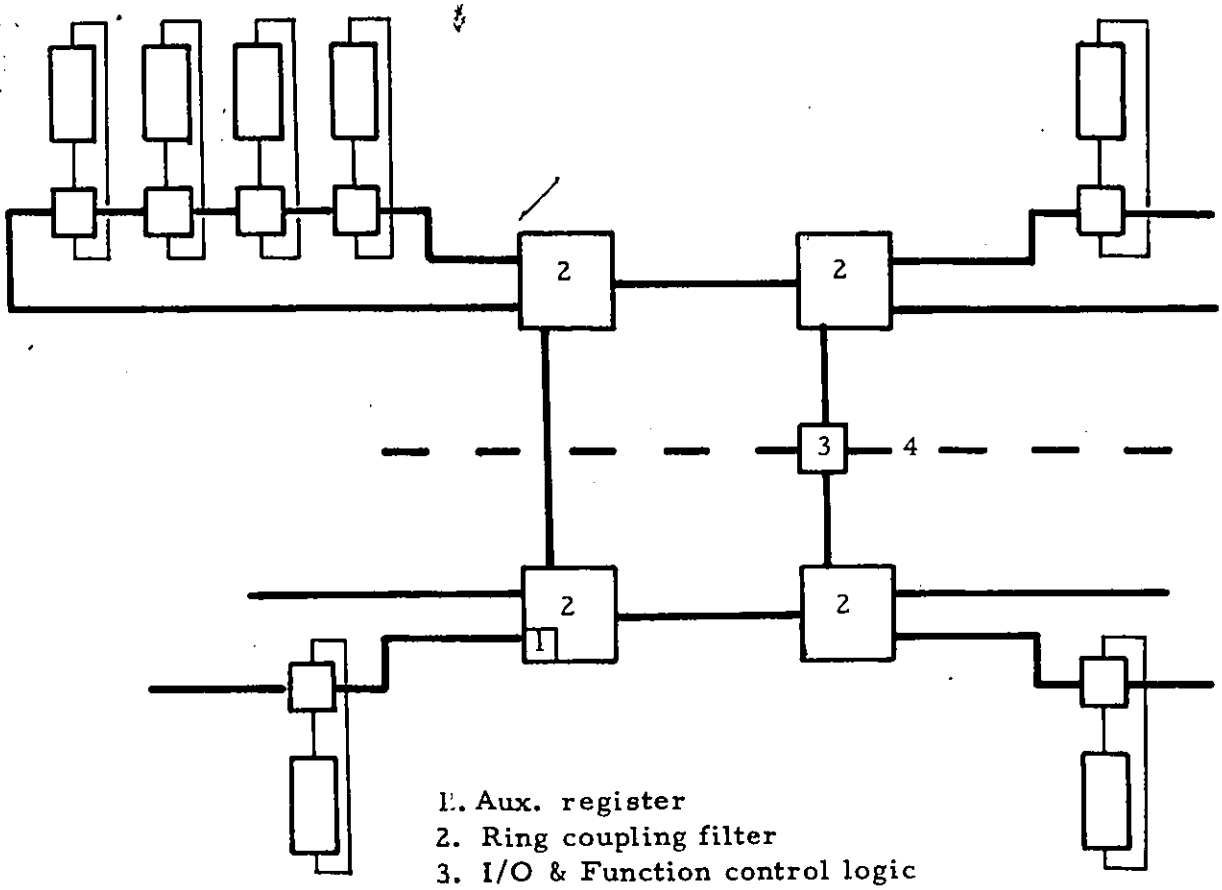
$$T_{(max)} = K \cdot W_m M_r t_c = K \cdot W_m \cdot (M_n / R_n) \cdot t_c$$

where $K > 1$

and R_n : the total No. of rings.

M_r : No. of memory units on a ring.

The improvement in sorting time is: R_n/K.



1. Aux. register
2. Ring coupling filter
3. I/O & Function control logic
4. Bus for more complicated system

Fig. 17 Four ring sorting system.

T(max) Simple ring	T(max) Complex configurations	Rn/K
55	45	1.22
50	41	1.22
75	63	1.19
55	55	1.0
60	50	1.2
45	58	1.12

Table 1. Values of the improvement factor: Rn/K.

Values for $Rn/_K$, shown on Table 1 can be determined by the simulation of some single and double ring configurations.

A random selection words was first stored in the system and sorted according to one descriptor set. Then the descriptor bits were changed and the number of clock pulses for the completion of the resorting process was determined. The flow chart and the programs are given in Appendix B. It can be seen that the value of K depends on the initial structural complexity of the file.

The memory system can be expanded by coupling the ring shown in Fig. 17 into a further ring segmented bus. The resulting ring can be coupled into another, and so on, depending upon the size of the overall system desired. In large systems, the designer has the choice of either employing a large number of descriptor bits and increasing the word length, or using the same descriptor for several memory segments. The rings are coupled by "ring control segments", which function as part of a major ring much as do the memory segments in a minor ring. The coupling filter is also similar, except that there are three inputs and three outputs, rather than two of each as in a minor ring. Thus, in an integrated realization, ring control segments can be used as memory segments, with one input and one output left unused. As in the case of the memory coupling filter, the ring coupling filter can route the data from any input to any output as determined by the descriptors.

Memory Segment

Two types of memory segment have been designed, an elementary static logic unit and a more flexible dynamic unit where

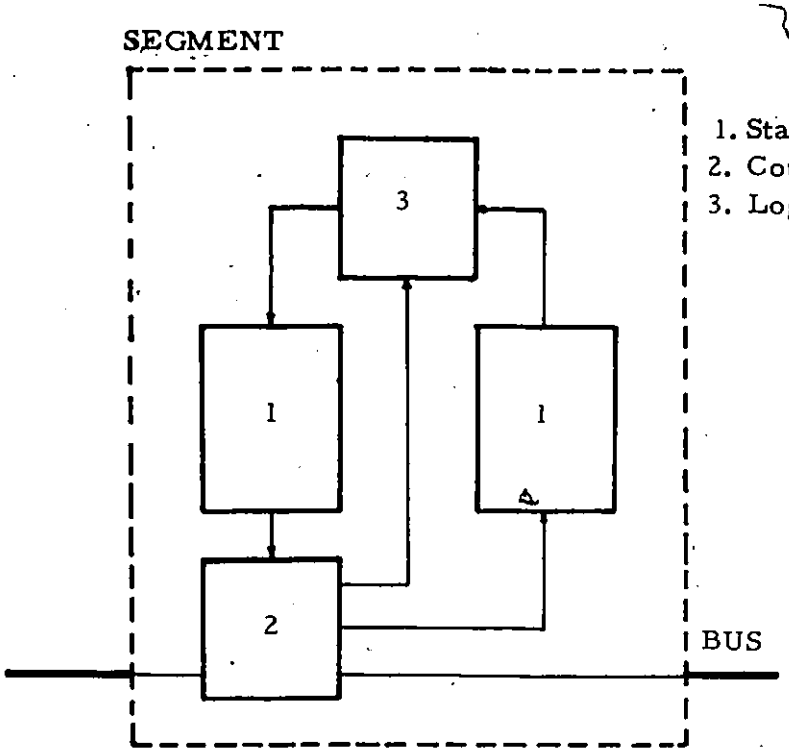


Fig. 18 Static memory unit

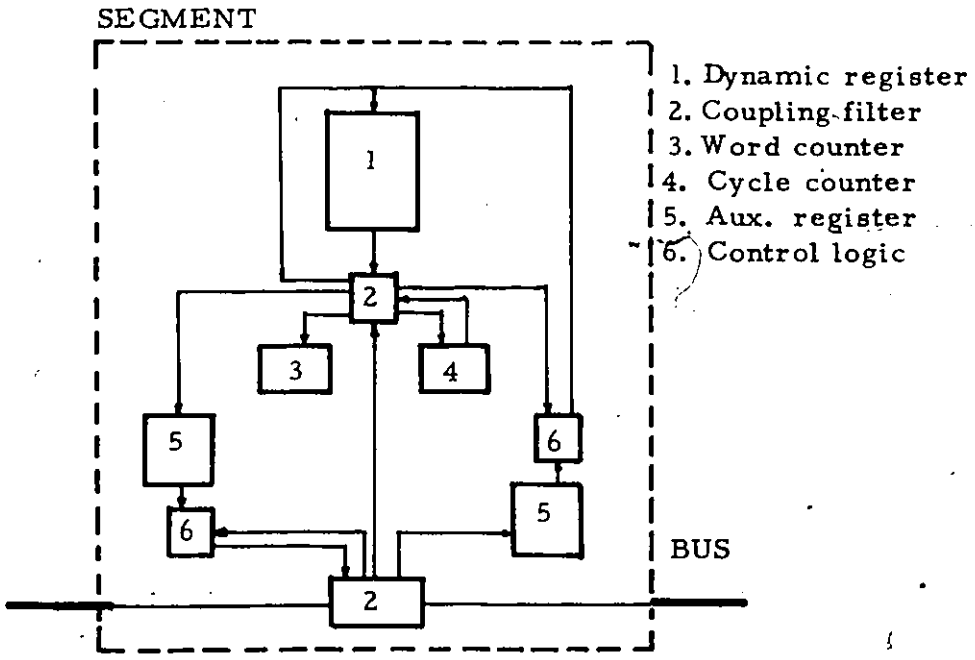


Fig. 19 Dynamic memory unit

the storage can be in a dynamic MOS shift-register, or a disk. Both units can be used either as a memory segment or a ring control segment, depending upon the number of connected bus inputs and outputs. The static-logic unit is shown in Fig. 18 . It consists of two static registers (block No. 1), the one on the left an output register and the one on the right an input register. A (coupling filter) unit in block 2 communicates between the registers and the bus. It allows words containing the descriptor for the segment to enter the input register, and outputs words from the output register into empty spaces on the bus. To control its next function, it also counts the words entering the input register. When the output register is empty, this is detected by the control logic and status information is transmitted to "Central Control". When the resort operation is complete, the words in the input register can be shifted through block 3 into the output register under the control of the counter in block 2.

Dynamic Memory Unit

The inefficient use of storage space in the static logic system of Fig. 18 is overcome in the system of Fig. 19 . Here a large memory (block 1) is used with two auxiliary registers (block 5), which become the input and output registers. These auxiliary registers are small, static logic FIFO stack memories, which are coupled to the bus through a coupling filter in a way similar to the input and output registers of Fig. 18 . The main memory of block 1, can be any read-write memory. (In the description below the main memory is a dynamic shift register.)

Note the similarity between entering a word into a dynamic shift register and a bus. In both cases, new information is introduced into

empty carriers or spaces in the stream and words with the appropriate descriptors are removed from the stream.

The dynamic memory unit operates in the following manner:

During the first cycle of the main memory, counter B (block 4) will be incremented every time a word passing the control unit (upper block 2) does not match the descriptors for the segment. When the first cycle is finished, the number in counter B indicates how many words will leave this segment during the process of resorting. Whenever space is available in the output auxiliary register, words which do not match the descriptor for the segment will be transferred there. During this operation, there is another transfer, words arriving from the bus enter the input auxiliary register through the control unit (block 2) and are transferred into the empty spaces of the main memory.

Design of the Dynamic Memory Module

To demonstrate the principles and characteristics of the module, a model was designed. A flow chart (Fig. 20) illustrates the algorithm implemented in hardware for the sorting function.

There are basically two comparator units in the module (Fig 21) They are similar; one controls the input of matching words from the bus to the module and the other controls the output of unmatching words from themodule to the bus. These two comparators are interconnected; if a word must be written on the bus but an empty carrier is not available, the word can still be written if the word on the bus is read by the same module. The module exchanges an unmatching word for a matching one. To perform such an qperation, which is a

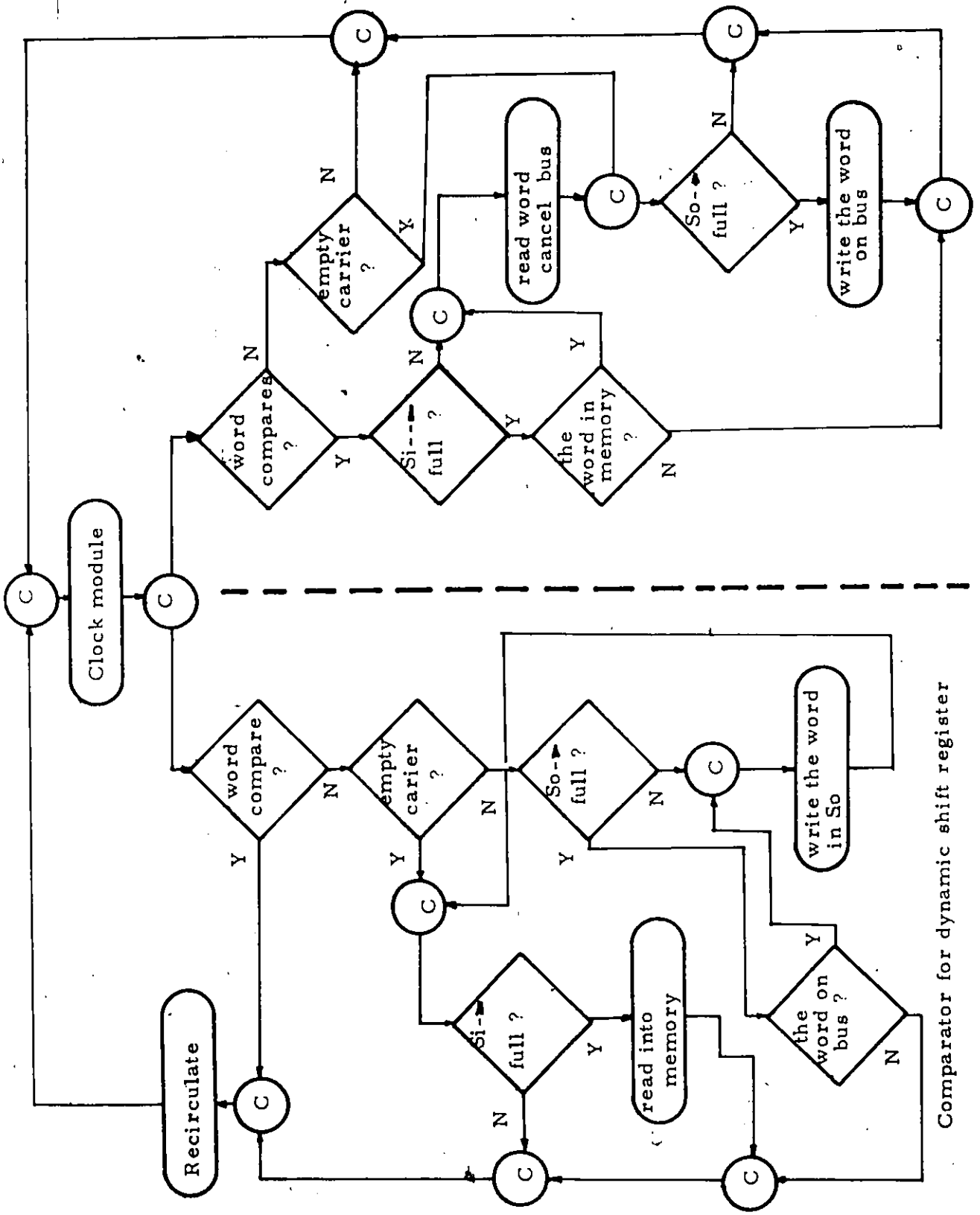
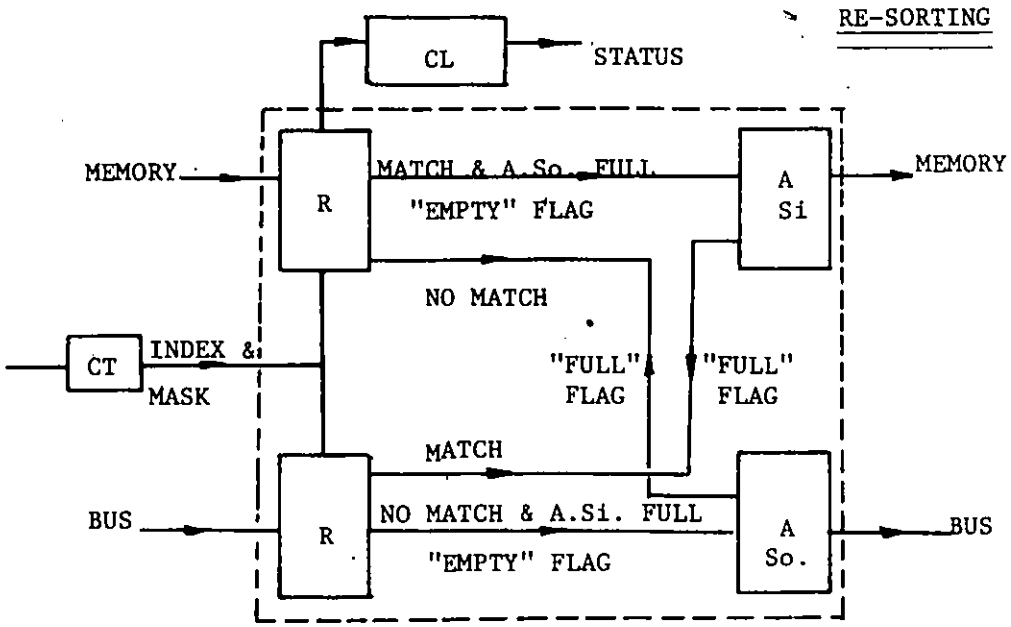


Fig. 20 Algorithm for the Coupling Filter.

Fig. 21 COUPLING FILTER FUNCTIONAL BLOCK DIAGRAM



R: CONTAINS ROUTING LOGIC' COMPARATOR & EMPTY DETECTOR

A,S,: OUTPUT & INPUT AUX. STACKS.

CT: CONTROL

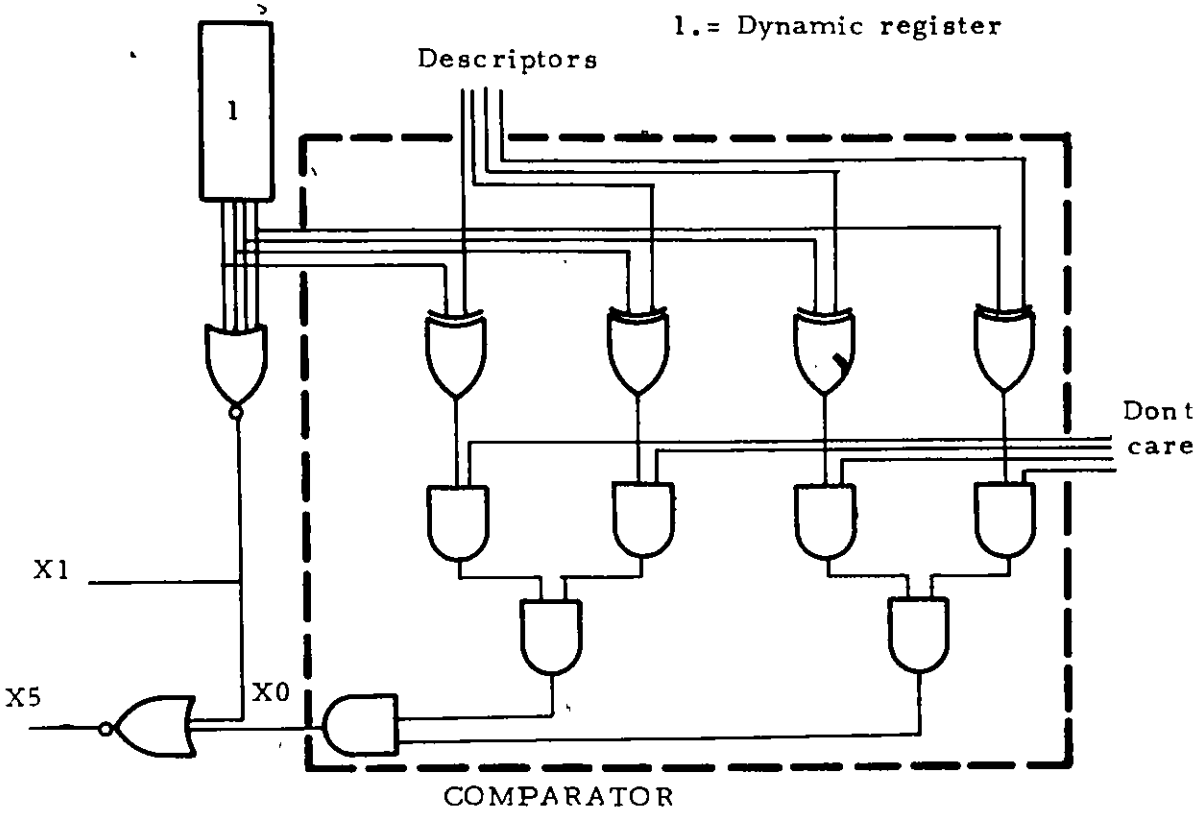
CL: COUNTER & LOGIC

requirement to minimize the resorting time, the two comparators must have a means of "Knowing" each other's state.

The function of the comparator, shown on Fig. 22 , is to indicate a match between the descriptor bits of the module and the corresponding bits in the word at the output of the dynamic register. An empty carrier is represented by the word '0000". To recognize an empty carrier a NOR gate is connected at the output of the dynamic register.

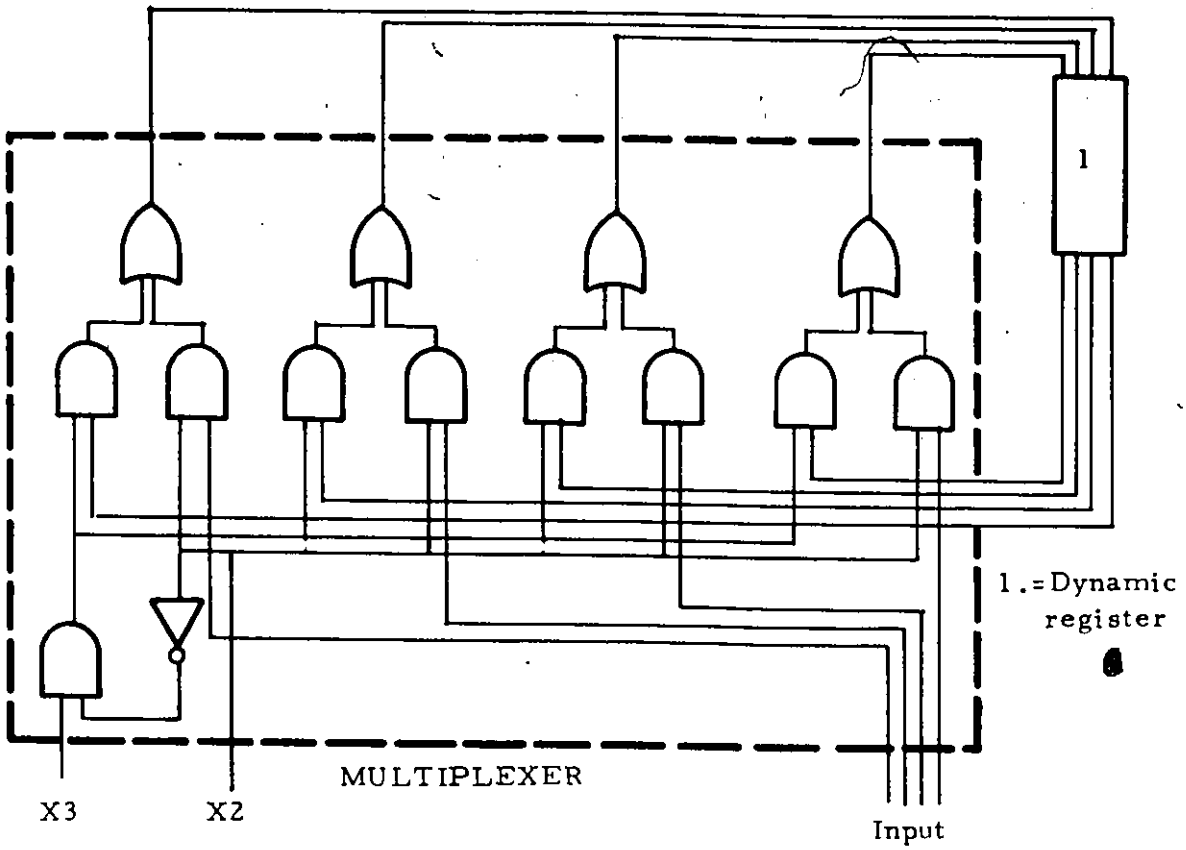
A multiplexer (Fig. 23) is required either to route a word to the input of the register or to read a new word. If neither of these can be done, then an empty carrier is introduced into the dynamic register.

Input and output stacks are 16 x 4 bit static registers. In order for the system to know if a word is stored in the register, there must be a flag bit for each word stored. The combinational circuit controlling the routing of the words was derived directly from the algorithm for the sorting function shown on Fig. 20 . Fig. 24 shows the algorithm for the word counter. The counter goes through one cycle when it counts the number of words in the dynamic register; hence it must have as many counts as there are words in the dynamic register. On the first cycle of the register the counter counts the matching words and empty carriers. On subsequent cycles it counts only the words taken out. When the counter reaches its highest count, all unmatching words are out of the module. The circuit diagram for the sorting module is shown on Fig. 25.



X1	X0
1___ Empty carrier	1__ No comparison
0 ___ Not empty	0___ Compares

Fig. 22



X2

1__ Read a new word

0___ Recirculate the word

X3

1__ Normal operation

0___ Read an empty carrier

Fig. 23

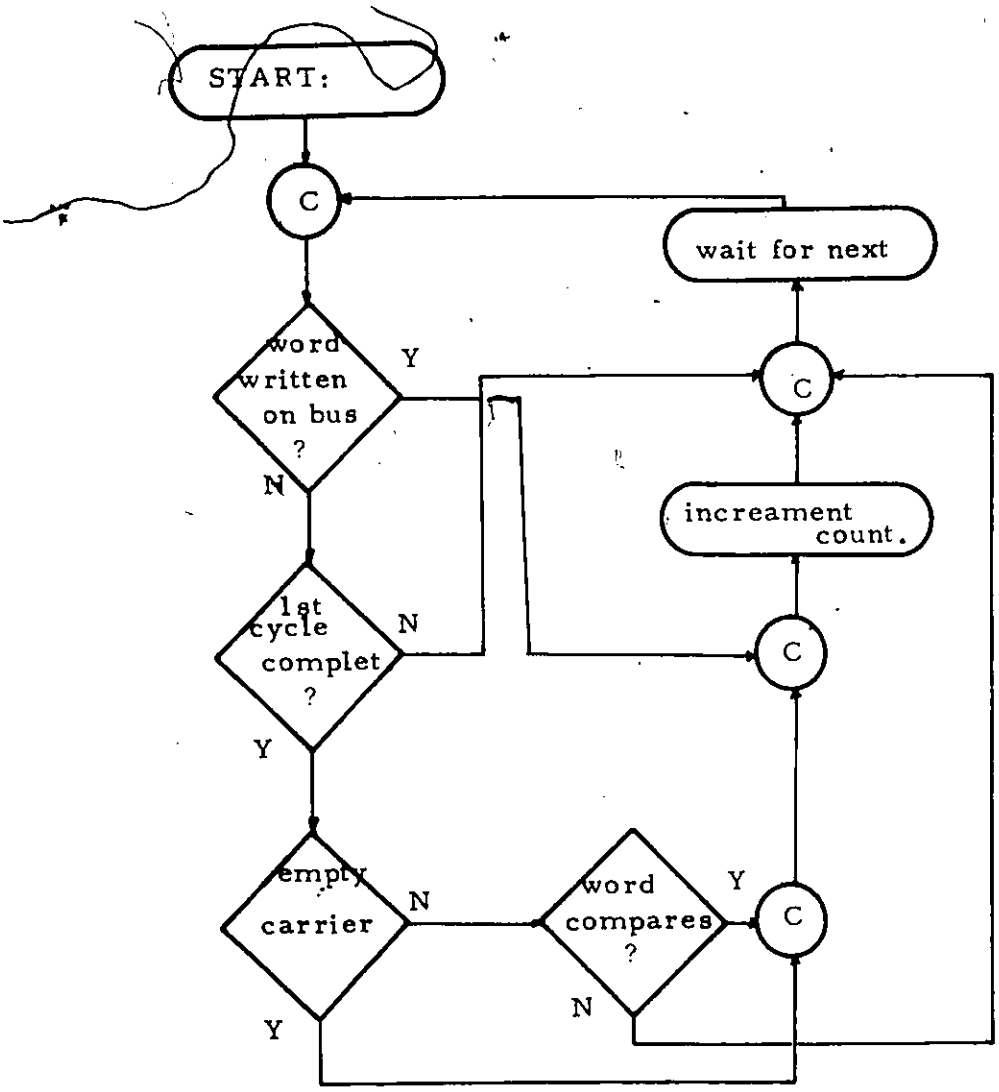
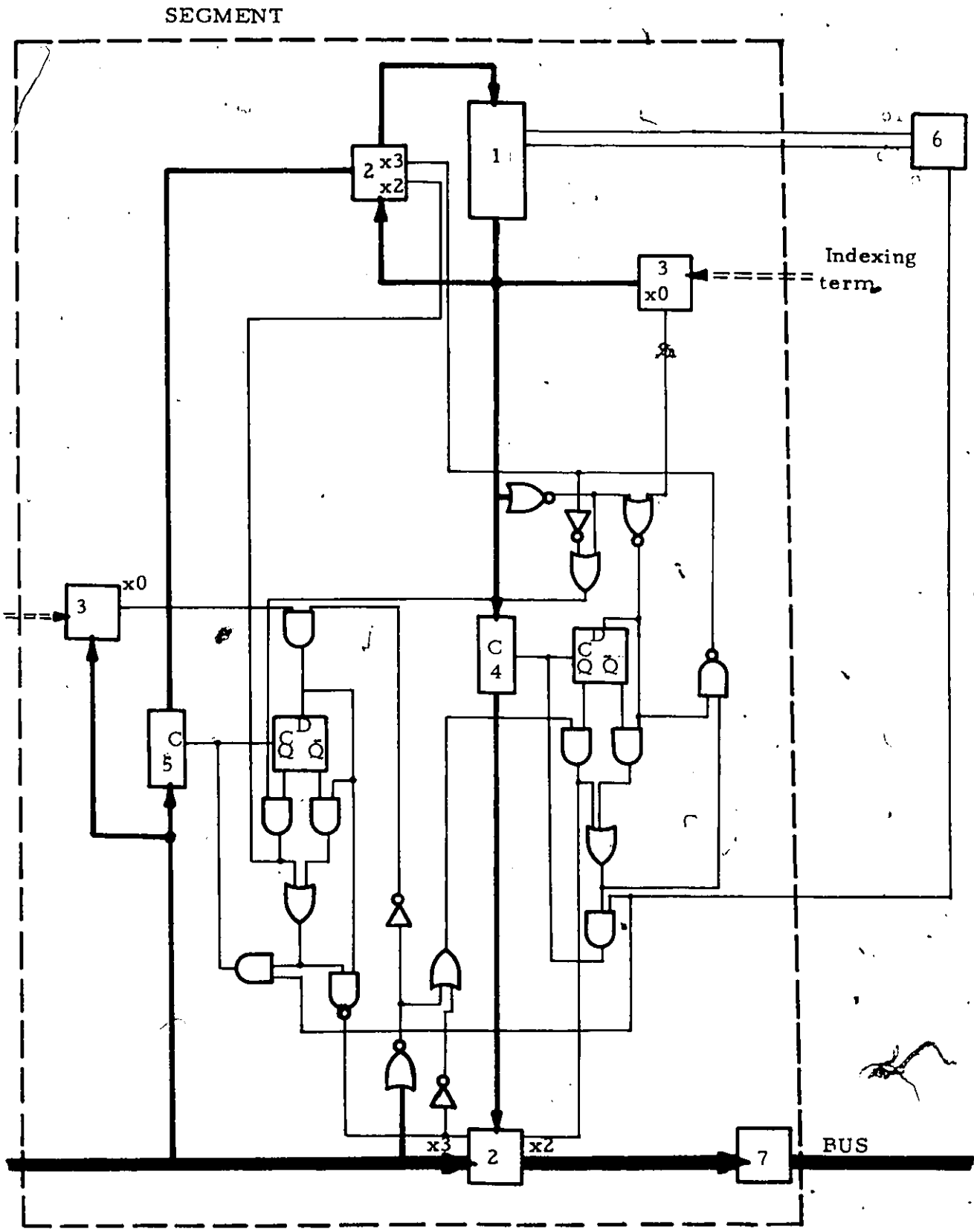


Fig. 24 Word Counter Algorithm.



1-Dynamic shift register;2-Multiplexer;3-Comparator:4-Output stack;
5-Input stack;6-Clock signal generator;7-One stage memory unit

Thus it has been shown that the sorting system described is feasible and the question arising is: would it be economical to build a system which is really an extra unit in a computer, or a system by itself, and which is used only for sorting and resorting of large amounts of data?

CHAPTER VI

GENERAL PURPOSE MEMORY SYSTEM

In chapter V a hardware implemented sorting system was described and its feasability was demonstrated, while leaving doubts as to its cost effectiveness.

A solution to the problem of economy is a flexible, multi-functional storage system, a system not only capable of performing the sorting function, but also able to provide direct and associative addressing facilities.

The system described in the previous chapter can be extended to a more flexible storage system with very few modifications. If each memory unit in Figure 15 is position addressable and is characterized by a unique address, then the system can operate as a direct addressed memory. Moreover, the simple ring structure of Figure 15 can be a pipe line memory, since modules are attached to a common bus and can be addressed separately in an interleaved fashion. Because the content addressable technique is used for sorting, the associative function can be easily implemented. Thus a multi-function storage system has several advantages over competitive types of memory organization.

Universal Memory Module

The memory unit described in the sorting system can now be called a universal memory module, because of the different operating modes. The sorting mode has been already considered in the previous

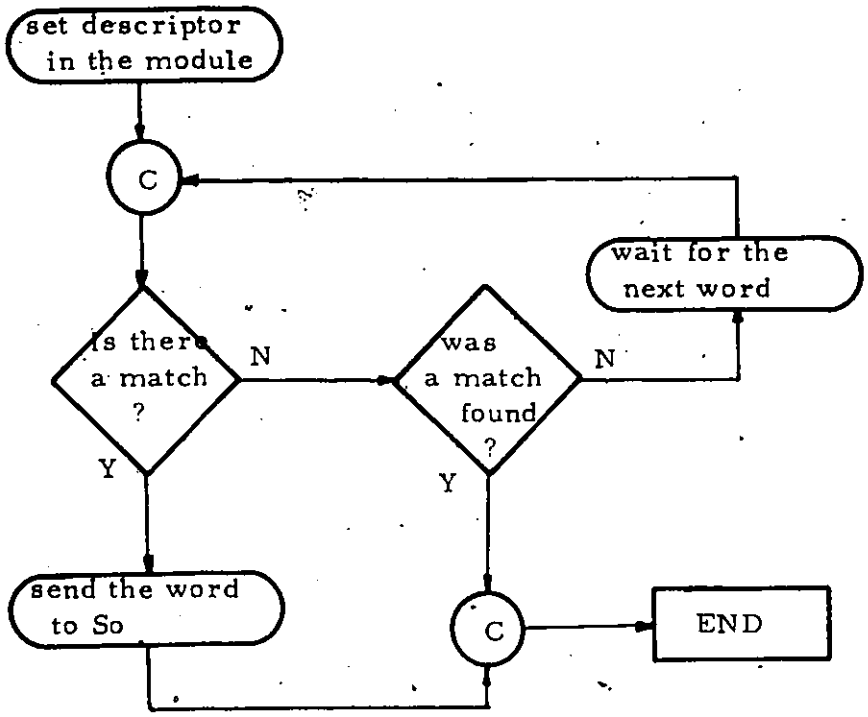


Fig. 26 Associative mode algorithm.

chapter. The flow chart for the algorithm used in the associative mode is shown on Figure 26. Whenever a matching word is found, it is sent to the bus with a non-destructive read-out. When the first matching word or a matching block of data is found, the operation is terminated. The word can then be sent to the processor or input/output unit. An important factor is that the search time is significantly reduced using the sorting system.

To operate in the direct address mode, the module must write on the bus the words whose addresses match. That is, the address determines the module and also the position of the word within that module.

Part of the address is compared with the cycle counter of the system and when it matches, the word is then at the output of the dynamic register of the addressed module.

Design of the Universal Memory Module

The circuits designed for the sorting system in chapter V can be modified to fit the requirement of other modes of operation.

To operate in the associative mode, the circuit that takes words out of the dynamic register must be modified. Whenever the word compares to the indexing term, then the flag bit becomes "true" and the output stack is clocked. This same signal informs the Central Control Unit that a word was found, whereupon this unit terminates the search. Then the circuit designed for the sorting mode writes the word on the bus. Note that a non-destructive read-out method is used

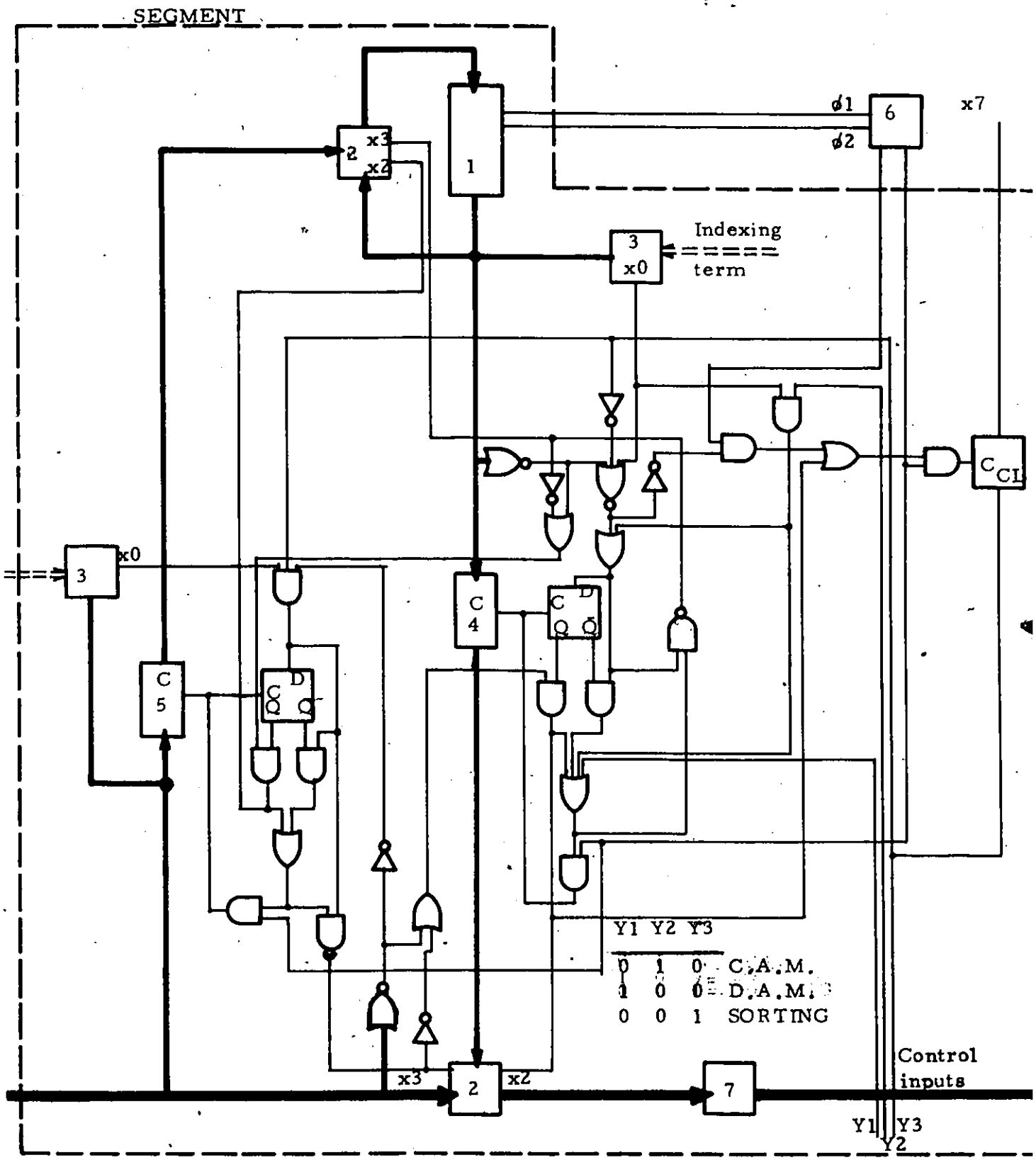


Fig. 27 Circuit diagram of the Universal Memory Module.

there but a destructive read-out can also be implemented.

The direct mode is simple to implement. When the cycle counter reaches the appropriate address, a signal sent to the addressed module replaces the comparator signal. The circuit used for the associative mode will respond to the latter signal and take the word to the output stack.

In this procedure the word was assumed to be needed right away. The procedure can also be arranged so that the word stays in the output stack until the Central Processor Unit asks for it.

A useful facility is the non-destructive read-out, which permits the CPU to process data obtained from the memory and return the result to the appropriate address only.

Figure 27 shows a circuit representing a Universal Memory Module. Note that if the size of the word is 4 x m bits then the circuit can be built on a 20-pin package using high density technology, e.g. MOS. This design demonstrates the feasibility of the project as well as the simplicity of the design.

System Improvement

One of the problems encountered in the system was the performance of the output stack. Before a word can be written on the bus, there must be a sufficient number of clock pulses to transfer the word to the output. Otherwise, even though there are empty carriers on the bus, the first word will enter the bus only after it reaches the bottom of the stack. A solution to this problem can be a specialized stack

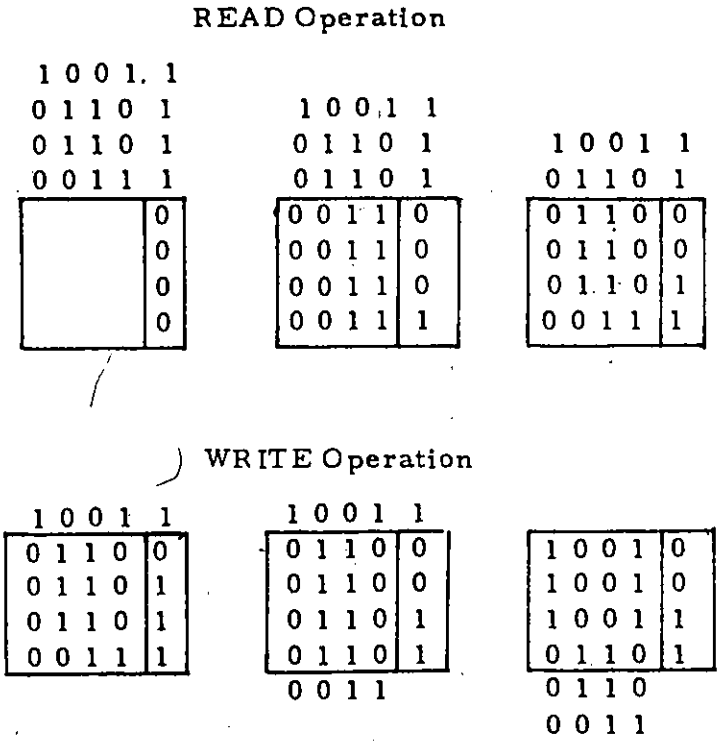
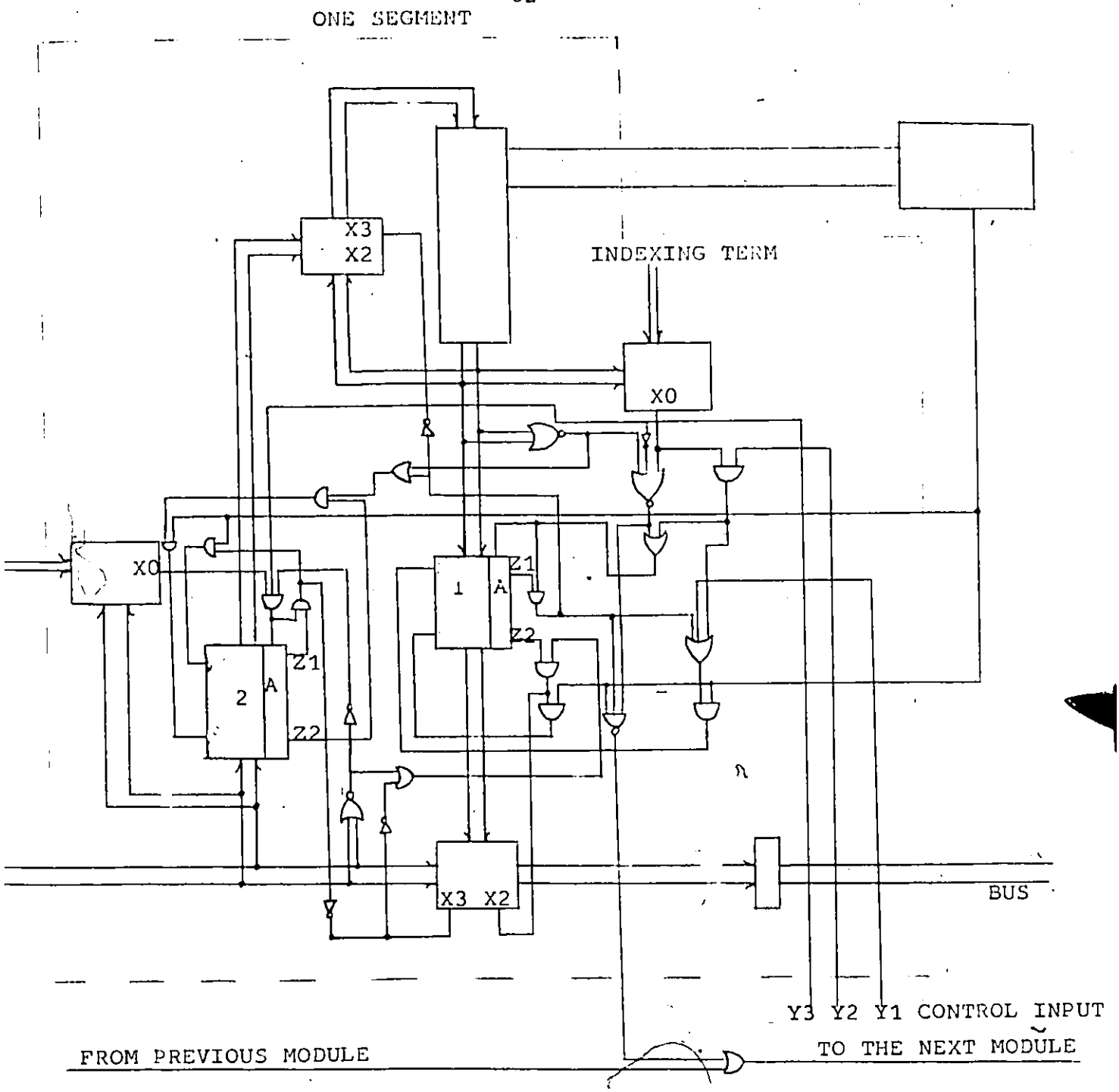


Fig. 28 Bubble-up register during read and write operations.



Z1 is the flag bit of the first stage.
Z2 is the flag bit of the last stage.
A indicates the flag bits for the whole register.
1 Input stack
2 Output stack

FIG. 29 Universal Memory Module using a race mode bubble-up register for the input and output stacks.

which reads a word and transfers it to the output as fast as possible. Thus this stack must fill from the bottom and not the top. For this reason it is called a bubble-up register.

Figure 28 illustrates the operation of such a register, and Figure 29 shows the Universal Memory Module with the improvement discussed above.

For demonstration purposes a model of the Universal Memory Module was built using available technology. Details of the model are given in Appendix C

CHAPTER VII

CONCLUSIONS

The previous chapters demonstrate the feasibility of designing a general purpose modular memory system which can operate in either a direct address or associative address mode. In order to improve access speed in the associative mode, it also can provide a sorting and re-sorting facility.

Each module of such a memory can independently perform in any of the above modes according to instructions from a central control. Thus, at any time, one part of the memory system can operate in a direct address mode, while another part can be operating in an associative mode. At the same time, a third part of the memory can re-sort data previously stored.

It has been shown that the basic elements of the system are storage devices and coupling filters. The data is always regarded as being in a stream, so the system is suitable for such economical storage devices as dynamic MOS shift registers, charge coupled devices and "floppy disks". Because of the dynamic nature of the system, it would seem reasonable to use dynamic MOS for the coupling filters. However, although experimental modules have been constructed, no LSI design has been done and it may prove that a modern bipolar technology, such as I^2L, would be equally suitable. This opens a fertile field for future work.

Other areas for research include incorporating error self-correction into the system [13a] and the application of the memory to various kinds of data processing system. For example, if the memory

system is to be operated in several modes at the same time, the data associated with the various modes must be identified by a suitable flag system. However, such a flag system can not be designed without a set of system requirements and specifications.

Even without an overall data processing system requirement, it is possible to identify some of the properties of the memory system. It is flexible, it is suitable for expansion to large systems and its final cost is primarily the cost of the memory devices. The coupling filters are relatively simple integrated devices and the dynamic nature of the storage devices ensure a cost-per-bit effective system.

APPENDIX A

CONTENT ADDRESSABLE MEMORIES

1. Intel 3104

   Technology: Schotty bipolar

   Size : 4 x 4

   Speed : 15 msec search, 30 msec write

   Package : 24 pin dip

2. Signetics 8220

   Technology : Bipolar

   Size : 4 x 2

   Speed : 20 msec associate, 30 msec read, 45 msec write.

   Package : 16 pin dip

3. Solic State Scientific SCL 5533

   Technology : CMOS

   Size : 8 x 8

   Speed : Read/Write access 150 msec, Read/Match cycle 200 msec Interrogate match for one bit 180 msec Interrogate match for eight bits 110 msec

   Package : 40 pin dip

4. 'Texas Instruments TMS 4000

Technology:	MOS P		
Size	:	16 x 8	
Speed	:	Interrogate acces time	50 nsec
		Read access time	30 nsec
		Write time	60 nsec
Package	:	40 pin dip	

Note: Texas Intruments and Fairchild have produced CAMs in the past but these have been discontinued.

A rescent publication reveals the development of a 128-bit MOS associative memory. The memory array can be integrated on a 1.6 x 2.9 mm chip using standard silicon gate MOS technology. Predictions of speed are 10 nsec for match and read access time and 25 nsec for write operation. It is possible to build a low cost associative memory array organized as 256 words of 256 bits each, which is TTL compatible (using a specialized buffer interface) and will operate with a 100 nsec cycle time. [20]

APPENDIX B

SIMULATION OF THE SORTING SYSTEM

The following is the flow chart for the computer simulated sorting system.

To estimate the time of the resorting operation a counter counts the number of steps or cycles indicated by the algorithm to complete the operation. This count is equivalent to the shifting of the segmented bus.

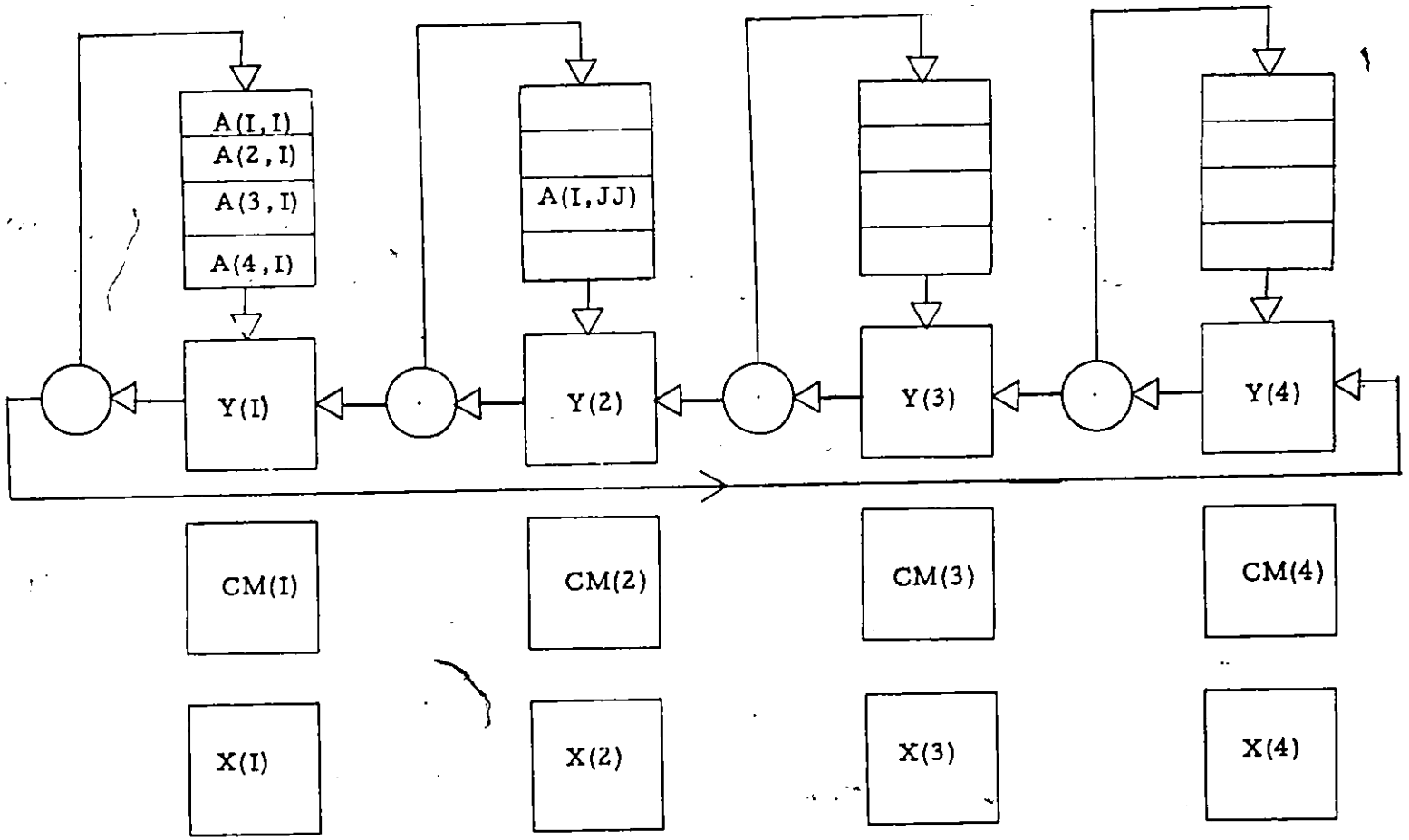


Fig. B1 Simulated system.

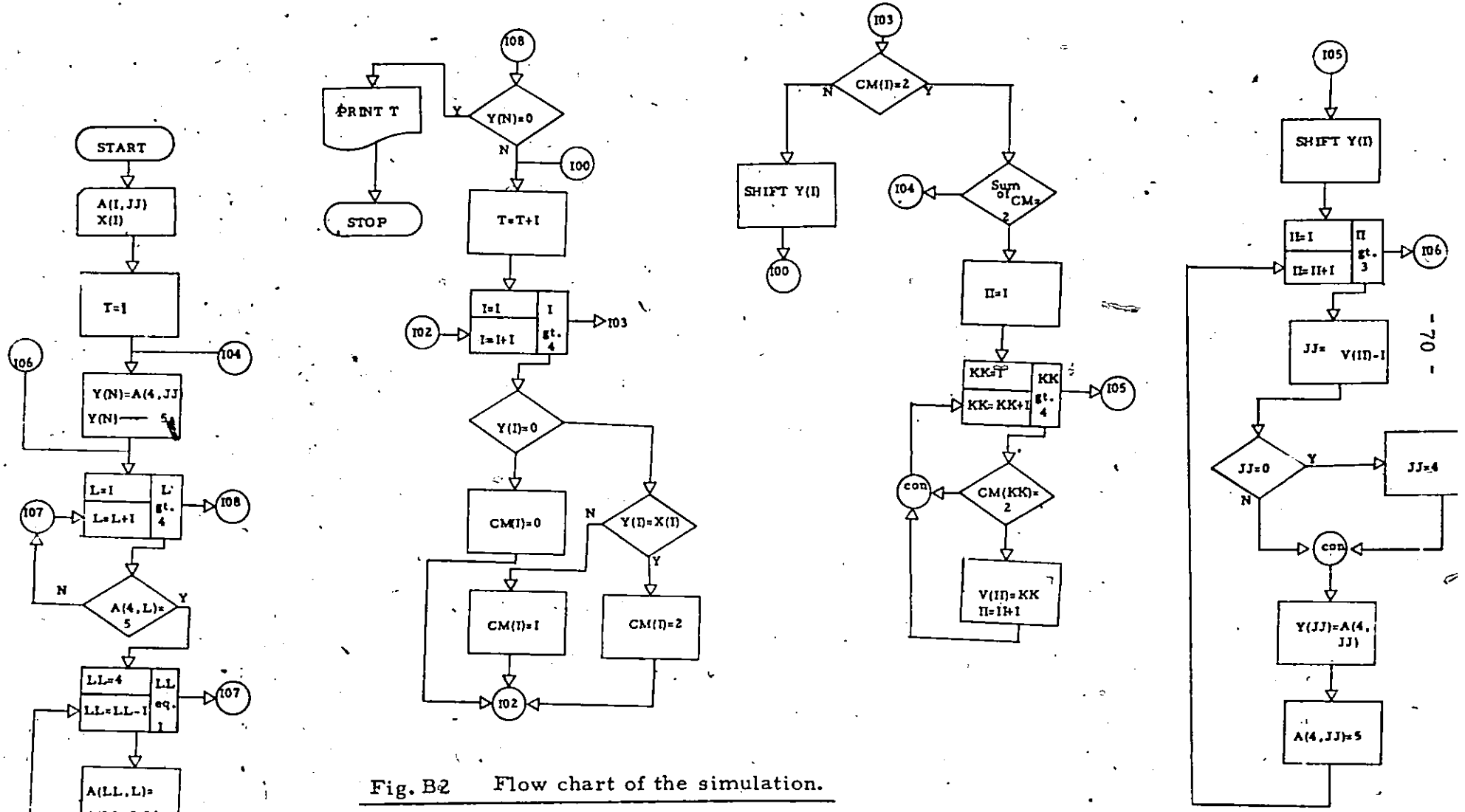


Fig. B2 Flow chart of the simulation.

SIMPLE RING CONFIGURATION

```
      $JOB  ACCT-NUM,'GLANZ.Z.'
 1           INTEGER A(4,16),X(16),Y(16),CM(16),V(16),C(1
 2          READ(5,10)((A(I,JJ),JJ=1,16),I=1,4)
 3       10 FORMAT(16(I3))
 4          READ(5,11)(X(I),I=1,16)
 5       11 FORMAT(16(I3))
 6           T=1
 7     1556 FORMAT(' ',I2,5X,I2,5X,I2,5X,I2,5X,I2,5X,I2,5X,I2,5X,I2,5X,I2,5X,I2,
            15X,I2,6X,I2,6X,I2,5X,I2,6X,I2,6X,I2,6X,I2)
 8     1557 FORMAT(' ',I2,5X,I2,5X,I2,5X,I2,5X,I2,5X,I2,5X,I2,5X,I2,5X,I2,5X,I2,5X,
            15X,I2,6X,I2,6X,I2,6X,I2,6X,I2,6X,I2,6X,I2)
 9       104  DO 90 IT=1,16
10            Y(IT)=A(4,IT)
11            A(4,IT)=5
12        90 CONTINUE
13       106  DO 60 L=1,16
14            IF(A(4,L).EQ.5)GO TO 1
15            GO TO 60
16          1 LL=4
17          2 IF(LL.EQ.1)GO TO 61
18            A(LL,L)=A(LL-1,L)
19            LL=LL-1
20            GO TO 2
21         61 A(1,L)=0
22         60 CONTINUE
23            MRA=0
24            DO 300 IC=1,16
25            MRA=Y(IC)+MRA
26       300  CONTINUE
27            IF(MRA.EQ.0)GO TO 3
28           IF(T.LT.10)GO TO 4
29           IF(T.GT.15)GO TO 4
30           PRINT,'STEP NO.',T
31           PRINT 1555
32     1555 FORMAT(' ','A(1)',3X,'A(2)',3X,'A(3)',3X,'A(4)',3X,'A(5)',3X,
           1'A(5)',3X,'A(7)',3X,'A(8)',3X,'A(9)',3X,'A(10)',3X,'A(11)',3X,
           1'A(12)',3X,'A(13)',3X,'A(14)',3X,'A(15)',3X,'A(16)')
33            DO 1554 I=1,4
34            PRINT 1556,A(I,1),A(I,2),A(I,3),A(I,4),A(I,5),A(I,6),A(I,7),
           1A(I,8),A(I,9),A(I,10),A(I,11),A(I,12),A(I,13),A(I,14),A(I,15),
           1A(I,16)
35     1554 CONTINUE
36           PRINT,'**************************************************************
           1**************************************************************'
37            PRINT 1557,Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7),Y(8),Y(9),Y(10),
           1Y(11),Y(12),Y(13),Y(14),Y(15),Y(16)
38           PRINT,'**************************************************************
           1**************************************************************'
39          4  T=T+1
40             DO 50 I=1,16
41             IF(Y(I).EQ.0)GO TO 5
42             IF(Y(I).EQ.X(I))GO TO 6
43             CM(I)=1
44             GO TO 102
45           5 CM(I)=0
46             GO TO 102
47           6 CM(I)=2
48       102  CONTINUE
49        50  CONTINUE
50            DO 38 IS=1,16
```

```
51            IF(CM(IS).EQ.2)GO TO 7
52         38 CONTINUE
53            C=Y(1)
54            DO 51 I=1,15
55            Y(I)=Y(I+1)
56         51 CONTINUE
57            Y(16)=C
58            GO TO 4
59          7 DO 40 IP=1,16
60            IF(CM(IP).NE.2)GO TO 41
61         40 CONTINUE
62            GO TO 104
63         41 II=1
64            DO 52 KK=1,16
65            IF(CM(KK).NE.2)GO TO 200
66            V(II)=KK
67            II=II+1
68        200 CONTINUE
69         52 CONTINUE
70           IF(II.EQ.16)GO TO 9
71           DO 53 IC=II,16
72            V(IC)=0
73         53 CONTINUE
74          9 C=Y(1)
75            DO 201 IL=1,15
76            Y(IL)=Y(IL+1)
77        201 CONTINUE
78            Y(16)=C
79            DO 210 IR=1,16
80            IF(V(IR).EQ.0)GO TO 109
81            JJ=V(IR)-1
82           IF(JJ.EQ.0) JJ=16
83            Y(JJ)=A(4,JJ)
84            A(4,JJ)=5
85        210 CONTINUE
86         109 GO TO 106
87           5 PRINT,'COMPLIIED IN',I,'STEPS'
88            STOP
89            END
```

COMPLEX CONFIGURATION

```
      $JOB   ACCT-NUM,'GLANZ,Z...'
 1           INTEGER A(4,16),Y(16),X(16),YY(5),XX(4),C(16),CM(4),CMM(4),V(16,4)
 2           INTEGER FLAG(4)
 3           INTEGER T,CC,SUM,SSUM,SUMM
 4    1551   FORMAT(' ','A(1)',3X,'A(2)',3X,'A(3)',3X,'A(4)',40X,'A(5)',3X,
     1'A(6)',3X,'A(7)',3X,'A(8)')
 5    1553   FORMAT(' ',I2,6X,I2,6X,I2,6X,I2,42X,I2,6X,I2,6X,I2,6X,I2)
 6    1554   FORMAT(' ',I2,6X,I2,6X,I2,6X,I2,42X,I2,6X,I2,6X,I2,6X,I2)
 7    1555   FORMAT(' ',I2,68X,I2)
 8    1556   FORMAT(' ',I2,68X,I2)
 9           READ(5,44)((A(I,JJ),JJ=1,16),I=1,4)
10     44    FORMAT(16(I3))
11           READ(5,45)(X(I),I=1,16)
12     45    FORMAT(16(I3))
13           READ(5,46)(XX(I),I=1,4)
14     46    FORMAT (4(I2))
15           DO 222 I=1,4
16           DO 223 II=1,16
17    223    V(II,I)=0
18    222    CONTINUE
19           DO 224 I=1,5
20    224    YY(I)=0
21           T=1
22           DO 50 IT=1,16
23           Y(IT)=A(4,IT)
24     50    A(4,IT)=5
25    100    DO 60 L=1,16
26           IF(A(4,L).EQ.5)GO TO 111
27           GO TO 60
28    111    LL=4
29    102    IF(LL.EQ.1)GO TO 60
30           A(LL,L)=A(LL-1,L)
31           LL=LL-1
32           GO TO 102
33     60    A(1,L)=0
34           DO 70 L=1,4
35           IF(V(1,L).EQ.5)GO TO 71
36           GO TO 70
37     71    DO 72 LL=1,15
38     72    V(LL,L)=V(LL+1,L)
39           V(16,L)=0
40     70    CONTINUE
41           DO 499 I=1,4
42    499    FLAG(I)=0
43           SUM=0
44           DO 101 I=1,16
45    101    SUM=Y(I)+SUM
46           IF(SUM.NE.0)GO TO 122
47           SUMM=0
48           DO 103 I=1,5
49    103    SUMM=SUMM+YY(I)
50           IF(SUMM.NE.0)GOTO 122
51           SSUM=0
52           DO 104 I=1,4
53           DO 105 II=1,16
54    105    SSUM=SSUM+V(II,I)
55    104    CONTINUE
56           IF(SSUM.NE.0)GO TO 122
57           GO TO 1000
58    122    T=T+1
```

```
      IF(T.LT.20) GO TO 1562
      IF(T.GT.23) GO TO 1562
      PRINT,'STEP NO.',T
      PRINT 1551
      DO 1552 I=1,4
 1552 PRINT 1553,A(I,1),A(I,2),A(I,3),A(I,4),A(I,5),A(I,6),A(I,7),
     1A(I,8)
      PRINT,'*****************************************************
     1*****************************************************'
      PRINT 1554,Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7),Y(8)
      PRINT,'*****************************************************
     1*****************************************************'
      PRINT 1555,YY(1),YY(2)
      DO 1557 I=1,4
 1557 PRINT 1556,V(I,1),V(I,2)
      LL=4
 1559 PRINT 1556,V(LL,4),V(LL,3)
      LL=LL-1
      IF(LL.EQ.0)GO TO 1558
      GO TO 1559
 1558 PRINT 1555,YY(4),YY(3)
      PRINT,'*****************************************************
     1*****************************************************'
      PRINT 1554,Y(16),Y(15),Y(14),Y(13),Y(12),Y(11),Y(10),Y(9)
      PRINT,'*****************************************************
     1*****************************************************'
      L=4
 1561 PRINT 1553,A(L,16),A(L,15),A(L,14),A(L,13),A(L,12),A(L,11),A(L,10)
     1,A(L,9)
      L=L-1
      IF(L.EQ.0)GO TO 1562
      GO TO 1561
 1562 DO1 I=1,4
      IF((YY(I)/10).EQ.XX(I))GO TO 2
      CMM(I)=1
      GO TO 1
    2 CMM(I)=2
    1 CONTINUE
      DO 3 I=1,4
      IF(CMM(I).NE.2)GO TO 3
      DO 1111 JC=1,16
      IF(V(JC,I).EQ.0)GO TO 112
 1111 CONTINUE
      GO TO 3
  112 V(JC,I)=YY(I)
      YY(I)=0
    3 CONTINUE
      CC=YY(1)
      DO 4 I=1,5
      IF(I.EQ.5)GO TO 5
      YY(I)=YY(I+1)
      GO TO 4
    5 YY(I)=CC
    4 CONTINUE
      DO 6 I=1,16
      IF(Y(I).EQ.X(I)) GO TO 7
      C(I)=1
      GO TO 6
    7 C(I)=2
    6 CONTINUE
```

```
113           DO 8 I=1,4
114           IF(I.EQ.2)GO TO 9
115           IF(I.EQ.3)GO TO 10
116           IF(I.EQ.4)GO TO 11
117           IF((Y(1)/10).EQ.XX(1))GO TO 12
118           CM(1)=1
119           IF(C(1).NE.1)GO TO 8
120           IF(YY(1).NE.0)GO TO 8
121           YY(1)=Y(1)
122           FLAG(1)=1
123           Y(1)=0
124           GO TO 8
125      12   CM(1)=0
126           GO TO 8
127       9   IF((Y(5)/10).EQ.XX(2))GO TO 13
129           CM(2)=1
129           IF(C(5).NE.1)GO TO 8
130           IF(YY(2).NE.0)GO TO 8
131           YY(2)=Y(5)
132           FLAG(2)=1
133           Y(5)=0
134           GO TO 8
135      13   CM(2)=0
136           GO TO 8
137      10   IF((Y(9)/10).EQ.XX(3))GO TO 14
138           CM(3)=1
139           IF(C(9).NE.1)GO TO 8
140           IF(YY(3).NE.0)GO TO 8
141           YY(3)=Y(9)
142           FLAG(3)=1
143           Y(9)=0
144           GO TO 8
145      14   CM(3)=0
146           GO TO 8
147      11   IF((Y(13)/10).EQ.XX(4))GO TO 15
148           CM(4)=1
149           IF(C(13).NE.1)GO TO 8
150           IF(YY(4).NE.0)GO TO 8
151           YY(4)=Y(13)
152           FLAG(4)=1
153           Y(13)=0
154           GO TO 8
155      15   CM(4)=0
156       8   CONTINUE
157           CC=Y(1)
158           DO 16 I=1,4
159           IF(I.EQ.4)GO TO 17
160           Y(I)=Y(I+1)
161           GO TO 16
162      17   Y(4)=CC
163      16   CONTINUE
164           CC=Y(5)
165           DO 18 I=1,4
166           IF(I.EQ.4)GO TO 19
167           Y(4+I)=Y(4+I+1)
168           GO TO 18
169      19   Y(8)=CC
170      18   CONTINUE
171           CC=Y(9)
172           DO 20 I=1,4
```

```
173           IF(I.EQ.4)GO TO 21
174           Y(8+I)=Y(8+I+1)
175           GO TO 20
176        21 Y(12)=CC
177        20 CONTINUE
178           CC=Y(13)
179           DO 22 I=1,4
180           IF(I.EQ.4)GO TO 23
181           Y(12+I)=Y(12+I+1)
182           GO TO 22
183        23 Y(16)=CC
184        22 CONTINUE
185           DO 24 I=1,16
186           IF(I.LE.4)GO TO0498
187           IF(I.LE.8)GO TO 25
188           IF(I.LE.12)GO TO 26
189           IF(I.GT.13)GO TO 83
190           IF(C(13).EQ.2)GO TO 1007
191           IF(V(1,4).EQ.0)GO TO 24
192      1008 IF(FLAG(4).EQ.1)GO TO 2004
193           IF(C(I).NE.2)GO TO 24
194      2004 Y(16)=V(1,4)
195           V(1,4)=5
196           FLAG(4)=0
197           GO TO 24
198      1007 IF(V(1,4).NE.0)GO TO 1008
199           Y(16)=A(4,16)
200           A(4,16)=5
201           GO TO 24
202        83 IF(C(I).NE.2)GO TO 24
203            Y(I-1)=A(4,I-1)
204           A(4,I-1)=5
205           GO TO 24
206        26 IF(I.GT.9)GO TO 82
207           IF(C(9).EQ.2)GO TO 1005
208           IF(V(1,3).EQ.0)GO TO 24
209      1006 IF(FLAG(3).EQ.1)GO TO 2003
210           IF(C(I).NE.2)GO TO 24
211      2003 Y(12)=V(1,3)
212           V(1,3)=5
213           FLAG(3)=0
214           GO TO 24
215      1005 IF(V(1,3).NE.0)GO TO 1006
216            Y(12)=A(4,12)
217           A(4,12)=5
218             GO TO 24
219        82 IF(C(I).NE.2)GO TO 24
220           Y(I-1)=A(4,I-1)
221           A(4,I-1)=5
222           GO TO 24
223        25 IF(I.GT.5)GO TO 81
224           IF(C(5).EQ.2)GO TO 1003
225           IF(V(1,2).EQ.0)GO TO 24
226      1004 IF(FLAG(2).EQ.1)GO TO 2002
227           IF(C(I).NE.2)GO TO 24
228      2002 Y(8)=V(1,2)
229           V(1,2)=5
230           FLAG(2)=0
231           GO TO 24
232      1003 IF(V(1,2).NE.0)GO TO 1004
```

```
233           Y(8)=A(4,8)
234           A(4,8)=5
235           GO TO 24
236      81   IF(C(I).NE.2)GO TO 24
237           Y(I-1)=A(4,I-1)
238           A(4,I-1)=5
239           GO TO 24
240     498   IF(I.GT.1)GO TO 28
241           IF(C(I).EQ.2)GO TO 1001
242     1002  IF(FLAG(I).EQ.1)GO TO 2001
243           IF(C(I).NE.2)GO TO 24
244     2001  Y(4)=V(1,1)
245           Y(4)=V(1,1)
246           V(1,1)=5
247           FLAG(1)=0
248           GO TO 24
249     1001  IF(V(1,1).NE.0)GO TO 1002
250           Y(4)=A(4,4)
251           A(4,4)=5
252           GO TO 24
253      28   IF(C(I).NE.2)GO TO 24
254           Y(I-1)=A(4,I-1)
255           A(4,I-1)=5
256      24   CONTINUE
257           DO 777 I=1,16
258           IF(Y(I).NE.0) GO TO 777
259           IF(A(4,I).EQ.5) GO TO 777
260           Y(I)=A(4,I)
261           A(4,I)=5
262     777   CONTINUE
263           GO TO 100
264     1000  PRINT,'COMPLETED IN',I,'STEPS'
265           STOP
266           END
```

The above are two programs, one simulating the simple ring structure and the second simulating complex configuration. Fig. is a typical example of the computer output.

SIMPLE RING

STEP NO. n

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)	A(9)	A(10)	A(11)	A(12)
0	0	0	0	0	0	0	0	0	0	0	0
0	0	21	0	31	0	0	0	0	0	11	0
0	32	33	13	12	0	13	42	44	0	31	11
* 44	22	32	43	14	21	32	24	13	34	13	42
14	22	41	12	44	31	41	34	23	24	44	21

STEP NO. n+1

----------ETC.

COMPLEX CONFIGURATION

STEP NO. m

A(1)	A(2)	A(3)	A(4)
0	0	0	0
0	0	21	0
0	0	33	0
0	44	32	13
23	11	22	41

41 Aux.
42
44 stack

A(5)	A(6)	A(7)	A(8)
0	0	0	0
0	0	0	0
31	0	0	0
13	0	0	0
14	0	21	31

Small ring

32 44

Main loop

33 42

0
0
0
0

0
0
0
0

A(9)	A(10)	A(11)	A(12)
11	32	41	11
0	0	0	0
0	0	0	0

A(13)	A(14)	A(15)	A(16)
22	31	13	34
0	0	0	0

STEP NO. m+1---------ETC.

Fig. B3 Typical output of the computer simulation.

APPENDIX C

THE DEMONSTRATION MODEL

The demonstration model is similar to Figure 27 . There exist slight differences in the design. A MOS Dynamic Shift register was used to simulate the data bus. The direct address mode was not implemented due to its simplicity and another comparator was required to compare the address to the cycle counter. To read information into the model, special circuitry was included which reads the output of the four least significant bits of the cycle counter. This way, there is always a known number of matching words on the bus, an added convenience to users. Another difference is the cycle counter itself. Before the sorting operation is started, the count on the cycle counter must be 0 since the word counter has to know when the first cycle is completed. For this reason and also because no bounceless switch was available another flip-flop was added which starts the sorting operation whenever the counter reaches 0.

Except for the sorting mode control the circuit is the same. Figure C1 shows the circuit implemented for demonstration.

Components and technology

Because it was a convenient choice, a MOS dynamic register was used with a 256 x 4 bits storage area. All other components are standard TTL technology but this is not a prerequisite for the module. The input and output stacks are 4 X 1 bits parallel registers. The word size is four bits and all the words are considered to be

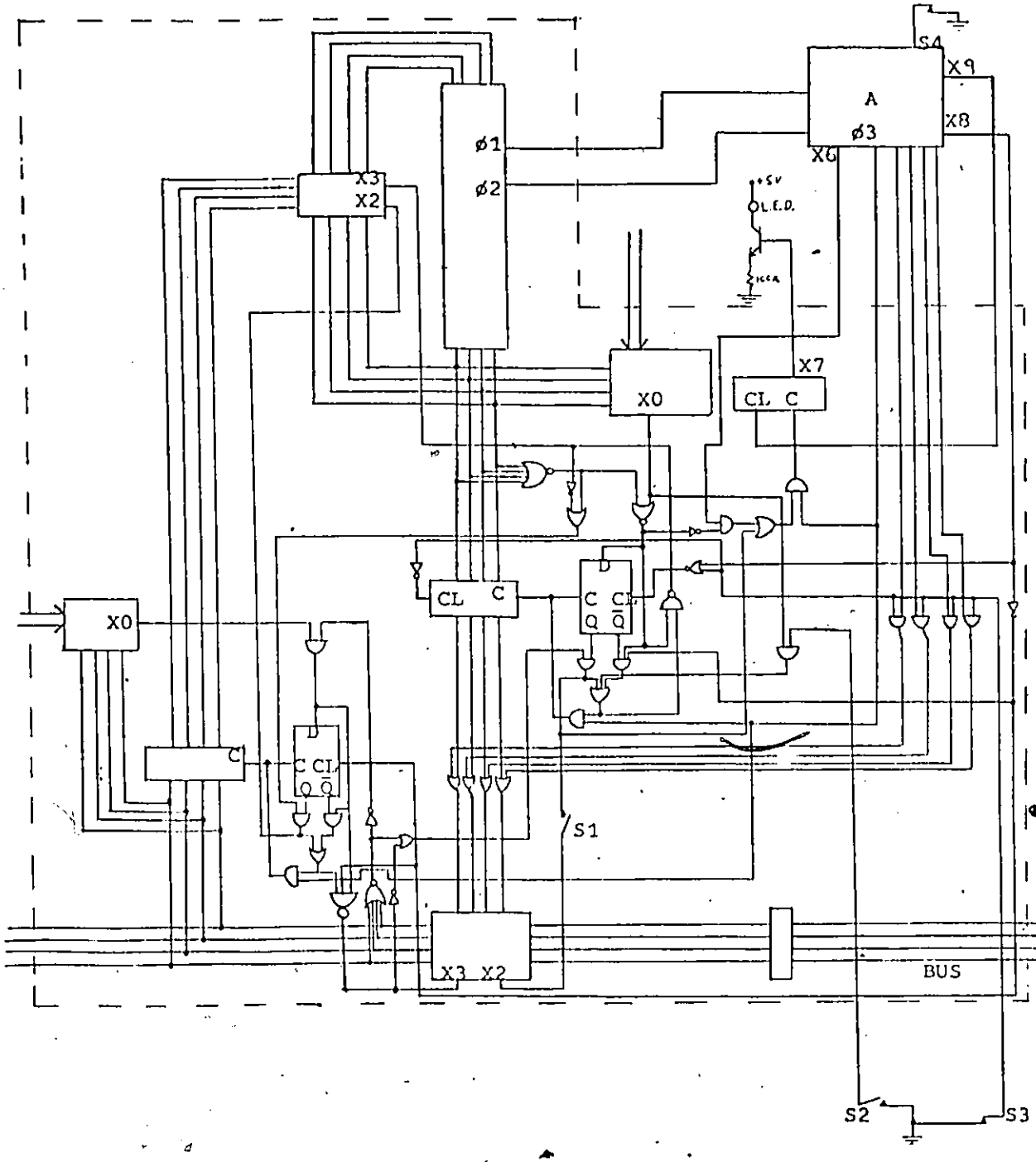
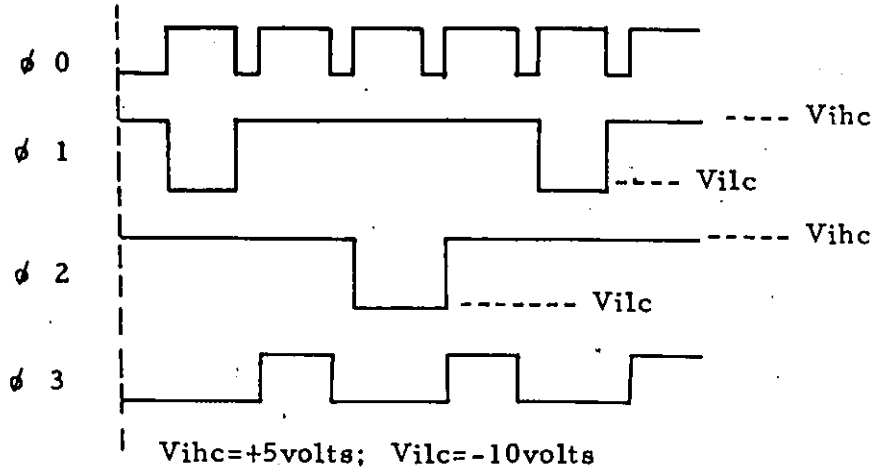


FIG. C1 This circuit diagram represents the model of the Universal Memory Module used for demonstration purposes.



Vihc=+5volts; Vilc=-10volts

NOTE:All other signals are TTL levels.

Timing diagram.

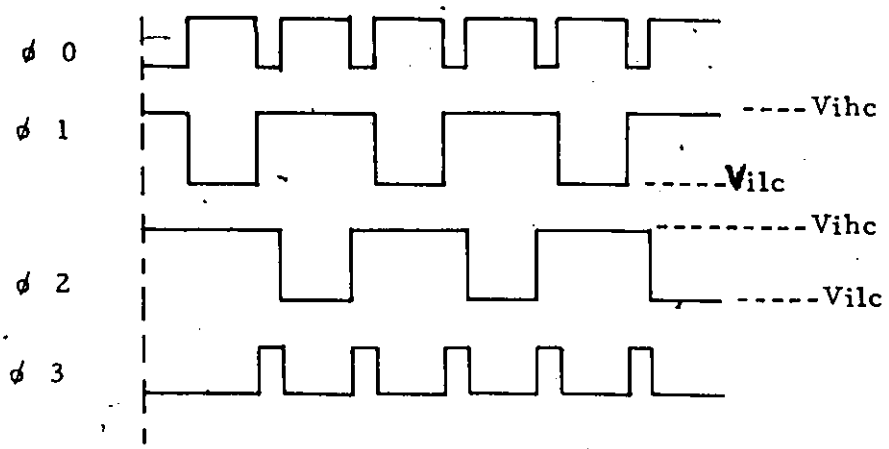


Fig. C3 Timing diagram.

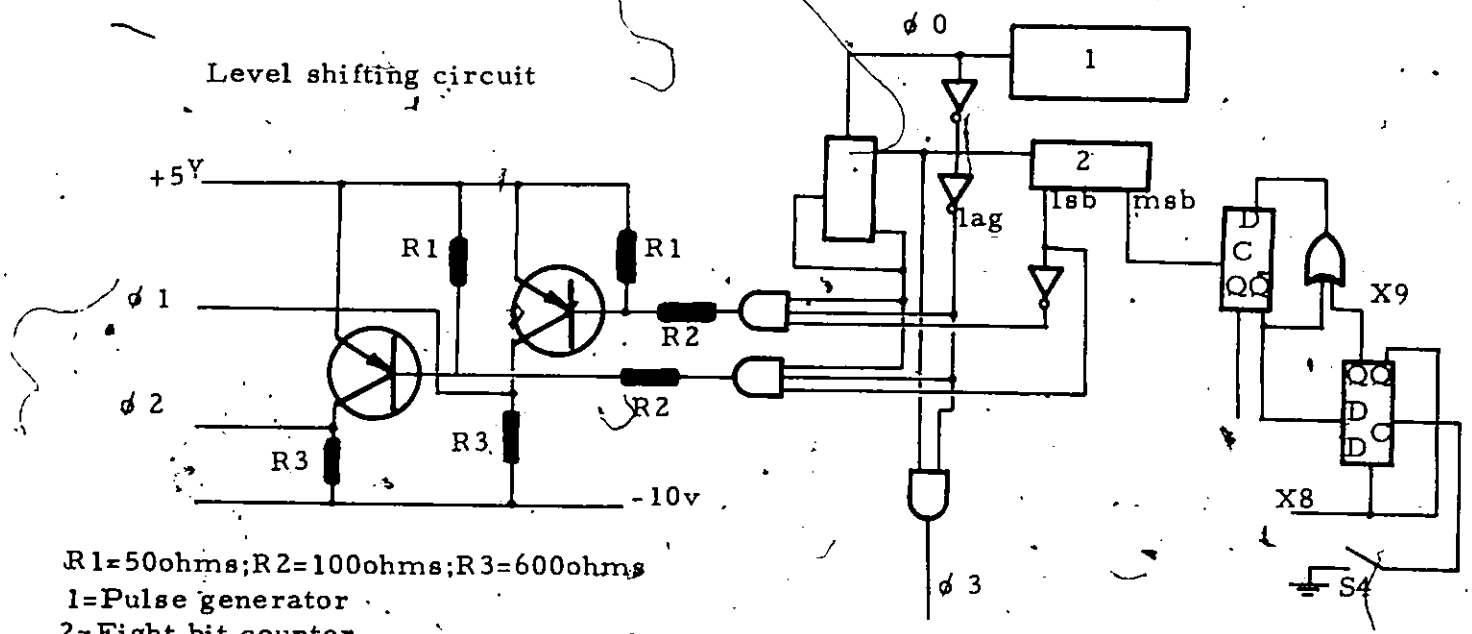
descriptors of imaginary data strings. Thus the indexing term of the module is also four bits.

Timing of the Operation

The sorting algorithm is fairly complex because many conditions must be checked before any decision is taken as to what route the word will follow. To implement these functions appropriate timing was required.

To operate MOS register must be clocked by two out of phase clock signals, ϕ_1 and ϕ_2, because the gate capacitance is the storage media, the data out clock delay is approximately 100 ns while the input set-up time is 30 ns. Another important characteristic is the on chip multiplexing technique of MOS register. This permits a data frequency of X MHz while the frequency of ϕ_1 and ϕ_2 is $\frac{X}{2}$ MHz. All MOS registers in the module were clocked by ϕ_1 and ϕ_2.

Because MOS registers are much slower than TTL, the input and output stacks must be clocked by a different signal, ϕ_3. Fig. C2 shows the timing of the input clock signal ϕ_0 as well as ϕ_1, ϕ_2, ϕ_3. When ϕ_1 reaches V_{ilc}, a new word appears at the output of the dynamic register 100ns later. A combinatorial logic circuit then determines the route of the word, based on the result of the comparison. If the word is to be recirculated to the input, then it must appear at the input 30ns before ϕ_1 reaches V_{ihc} because the dynamic register reads on the rising edge of ϕ_1 and ϕ_2. Whenever ϕ_1 or ϕ_2 reaches V_{ihc} ϕ_3 updates the results in the input and output stacks, based on the changes that have occurred.



R1=50ohms;R2=100ohms;R3=600ohms
1=Pulse generator
2=Eight bit counter

Fig. C4 Clock signals generator.

Due to the delays, the minimum clock pulse width is approximately 160ns, and this limits the maximum frequency. ϕ_3 can be narrower, namely 25ns, since TTL is faster than MOS. Thus a monostable multivibrator could be used to produce a narrow pulse whenever ϕ_1 or ϕ_2 reach V_{ihc} (Fig. C3). The frequency can be increased significantly using this method. But a monostable multivibrator was not available for the model. Fig. C4 shows the circuit used to produce the clock signals.

Test Results on the Model

The circuit was tested successfully. No race conditions were observed. Table C1 shows the different controls on the model.

S1	S4	S2	S3	FUNCTION
ON	OFF	ON	ON	Sort
ON	ON	ON	ON	Hold
OFF	ON	ON	OFF	Read words on bus
ON	ON	ON	ON	Recirculate the bus
OFF	OFF	ON	ON	Clear module
OFF	ON	ON	ON	Clear the bus
OFF	ON	OFF	ON	Associative mode

Table C1 - Instructions

There were 512 words in the bus register prior to a sort operation, obtained from the counter. An oxcilloscope was used to observe the behavior of the bus and the dynamic register in the module. As expected, when a sorting operation was started, matching words disappeared from the bus and appeared in the dynamic register. Since the order of the different combinations was known, it was easy to check if the right words were taken from the bus. Then resorting was successfully performed using a different indexing term. "Don't care" conditions were inserted in the indexing term to obtain more words in the dynamic register. When 10-- was used as the indexing term the dynamic register became full. Then the bus was filled again. and the indexing term was changed to 01--. This resorting operation required the transfer of 256 words out of themodule to the bus, and vice versa. However, the resorting time here does not represent a real system, since resorting was always completed after one cycle of the bus register.

In the associative mode, the module was filled with don't care conditions and the bus register was emptied. When an indexing term was given, and the operation started, a matching word was sent to the bus with a non-destructive read-out.

REFERENCES

CHAPTER I

(1) Speliotis, Dennis E., "Bridging the Memory Access Gap", AFIPS Conference Proceedings, 1975.

CHAPTER II

(2) Hughes, W.C., Lemmond, C.Q., Parks, H.G., Ellis, G.W., Possin, G.E., Wilson, R.H., AFIPS Conference Proceedings, 1975.

(3) Tompsett, N.F., Sealer, D.A., Bertram, W.J., Ségin, C.H., "Charge-Coupling Improves its Image", "Electronics" Jan. 1973.

(4) Amelio, G.F., "Charge-Coupled Devices For Memory Application", AFIPS Conference Proceedings, 1975.

(5) Staff Report, "Holography", Oprical Spectra, Jan. 1975.

(6) Gillis, A.F., Nelson, R.H., Hoffman, G.E., "Holographic Memories Fantasy or Reality", AFIPS Conference Proceedings, 1975.

(7) Dolgoff, Eugene, "Commercial Holography", Optical Spectra, March 1975.

(8) Ypma, I.E., "Bubble Domain Memory Systems" AFIPS Conference Proceedings , 1975.

(9) Cheng, H., Chen, T.C., Tung, C., "The Realization of Symmetric Switching Functions Using Magnetic Bubble Technology", AFIPS Conference Proceedings, 1973.

(10) Karp, H.R., "Magnetic Bubbles a Technology in Making", "Electronics", Sept. 1969.

CHAPTER III

(11) "MOS/LSI From Texas Instruments", October 1970.

(12) Jordan, W. F., "MOS Arrays", Electronics, Dec. 1968.

(13) "Intel Memory Design", August 1973.

(13 a) "Comparison of Memory Technologies" Spectrum Jan. 1975.

CHAPTER LV

(14) Flores, Ivan, "Computer Sorting", Prentice Hall, 1969.

(15) Rich, R. P., "Internal Sorting Methods Illustrated With PL-1 Program", Prentice Hall, 1969.

(15 a) Knuth, N., "Sorting and Searching"

(16) Kautz, W. H., "Cellular Logic in Memory Arrays" Transactions on Computers, August 1969.

CHAPTER V

(17) Thompson, P. M., Glanz, Z. H., "A Data Sorting System Using High Speed Bus", AFIPS Conference Proceedings, 1975.

(18) Champagne, C., -"A Bus Structure For A High Data Rate", submitted to AFIPS Conference Proceedings, 1975.

CHAPTER VI

(19) Glanz, Z. H., Charlebois, P., Thompson, P. M., "A Modular Multi-Functional Memory System For Fast Associative Processors" Digest of IEE Conference and Exposition, Sept. 1975.

APPENDIX A

(20) Lea, R.M., "MOS Associative Memory", The radio and electronic engineer, Vol 45, April 1975.

APPENDIX C

(21) Charlebois, P., "A Multi-Functional Data Storage System", submitted to Dept. of Electrical Engineering in partial completion of B.A.Sc. at University of Ottawa.