AN ABSTRACT OF THE THESIS OF

PETER TURNER RUX (Name) for the DOCTOR OF PHILOSOPHY (Degree)

in Electrical and Electronics Engineering (Major) presented on Feb 9, 1968 (Date)

Title: DESIGN AND EVALUATION OF A GLASS DELAY LINE CONTENT-ADDRESSABLE MEMORY SYSTEM

Abstract approved: Redacted for Privacy
D. L. Amort

The high cost of content-addressed memories (CAM) can be reduced significantly by using a circulating memory system. With the circulating system, comparison logic is shared by many memory locations to allow a reduction in cost.

Three data organizations are described and evaluated and one is chosen as the best for implementation when 20 MHz, 100 μsec glass delay lines are used.

The complete design of a CAM system is described which has a 2048 x 35-bit capacity and is capable of searching the entire contents in 100 μsec. Search criteria include equality and magnitude conditions. Operations include storing, fetching, and a variety of tagging functions.

The described CAM has been constructed and installed as an operating addition to a general-purpose serial computer, the

NEBULA. The CAM system has been a powerful addition to the NEBULA and a similar system could be a useful yet economical addition to any computer system.

Design and Evaluation of a Glass Delay Line Content-Addressable Memory System

by

Peter Turner Rux

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

June 1968

APPROVED:

Redacted for Privacy

Associate Professor of Electrical and Electronics Engineering

in charge of major

Redacted for Privacy

Head of Department of Electrical and Electronics Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented Feb 9, 1968

Typed by Clover Redfern for Peter Turner Rux

ACKNOWLEDGMENTS

It is a pleasure to acknowledge the willing assistance of Professor D. L. Amort and also all the people connected with the NEBULA project.

Special appreciation goes to my wife, Carolyn, for her constant encouragement and support.

It is also a pleasure to acknowledge the Office of Naval Research for providing financial support under contract #NONR-1286(11).

TABLE OF CONTENTS

LIST OF FIGURES

DESIGN AND EVALUATION OF A GLASS DELAY LINE CONTENT-ADDRESSABLE MEMORY SYSTEM

1. INTRODUCTION

1.1 Description and Capabilities of a Content-Addressable Memory

Most present-day computer memories are information storage systems which are capable of addressing information by means of its location in the memory. This type of memory is called a random-access storage system. However, there is another means of addressing information within a memory and that is by the information content with no regard to location. Such a memory has been called an associative memory or a content-addressed memory. In this paper, it will be referred to as a content-addressable memory or more simply, as a CAM.

For a word-organized memory, a CAM can address a word in memory on the basis of a comparison of that word with a reference word. For example, if a word is equal to the reference word, it will be addressed. However, since it is not known in advance where a word may be stored in the CAM, a CAM must be capable of comparing the reference word with all the stored words in order to find the word with the desired content.

To be able to address by content, the memory must be searched

and a word or words addressed on the basis of the criteria of a comparison of the stored words to the reference word. The simplest comparison criterion is that of equality but it is also useful to make magnitude decisions. Then it is possible to address words greater than the reference word for example.

After a CAM has completed a search and has addressed a word as satisfying the comparison criteria, then it will be desired to perform an operation on that word. The basic operations, of course, would be to fetch the addressed word, or to store new information into the addressed word or a portion of the addressed word. However, it is also possible to perform arithmetic operations or communication to neighboring words and ultimately it is possible to have an individual logic processor at each memory location with communication between processors and a central controller. The central controller can command a search and then for locations where the search is successful, an operation can be performed. This gives the CAM parallel processing capabilities for the search and for the operations.

1.2 Need for Content-Addressing

It is reasonable at this point to ask the question, "Why is a CAM desirable?" Content addressing can, in fact, be performed using a random-access memory by fetching a word from the memory, making a comparison with the reference word, and if the comparison

does not satisfy the comparison criteria, then new information is fetched and the comparison is repeated. This process is continued until either the information is found which was sought or until all of the information has been searched.

The "linear" search is a straight-forward approach which searches a table by first fetching the first word in the table, comparing, then fetching the second word, and so on until the table has been exhausted. If N items are in the table, N fetches and comparisons may be needed to find the word sought. Several techniques have been developed for searching a table of information which are improvements over the "linear" search.

If it is possible to order the information in a table, then a "binary" search may be used to improve on the searching speed of the "linear" search. In this type of search, one fetches the middle word from the table, determines whether the word sought is in the first half or the second half, then fetches the middle word from that half, and so on. The maximum number of fetches and comparisons necessary to search a table with N items is now $\log_2 N$.

Another method of speeding up an equality table search is called "hash" addressing. With this method, an item is stored in the table at an address which is computed from the item itself by application of an appropriate scatter function. If a different item is already stored at that address then the input item is stored at the next

consecutive unused location. Then to see if an item is in the table, the scatter function is applied to the item and the item is fetched from the computed address and is compared with the given item. If there is no match, then the next consecutive item is fetched and compared. Consecutive items are then fetched until the desired item is found. The speed of this type of search will depend on many factors but the efficiency of the scatter function is of prime importance.

These techniques allow tables to be searched rapidly in a random-access memory, but they are limited in their capabilities. For the "binary" search to be most effective, a rapid means must be available to compute the middle address of the group of data being searched. Also, it is essential that the table be ordered and maintained in order. If a new word is added to the table, then the table must be rearranged to maintain order. To use the binary search requires bookkeeping and calculations which can prove cumbersome.

The "hash" addressing method is a very good method of implementing rapid equality searches, but it too requires cumbersome address calculations. Also, "hash" addressing is not very flexible. Magnitude type searches are very difficult, as are modifications in the field of the search.

For problems where a search is required, the capability of content-addressing improves on a random-access memory by being faster, more flexibile, and simpler to program. The programmer

need only define what to search for and then command the search. Much of the bookkeeping necessary for the random-access search can be avoided.

In addition to the simple and fast searching, a CAM also provides a parallel processing capability which a random-access memory doesn't have. It is possible with a CAM to perform an operation on many memory locations simultaneously.

It is the ability of a CAM to perform parallel searches and parallel operations which makes it very useful for problems where parallel operations can be performed. A list of types of problems which would benefit from the availability of content-addressing would include:

1) Construction of symbol tables

2) Code conversions

3) Character and pattern recognition

4) Turing machines

5) List processors

6) Certain numerical problems

1.3 Methods of CAM Implementation

The basic requirement for implementation of a CAM system is to produce the capability of comparing the reference word with all stored words. This task can be performed by a random-access

memory by reading out one word at a time and comparing it with the reference word. So in the implementation of a CAM, it is desired to compare the reference word with all words simultaneously and then to operate on the word(s) which met the comparison criteria. To make a truly simultaneous comparison, it is necessary to have all of the information in the memory immediately available. This requirement can be satisfied by a static-memory system.

1.3.1 Static-Memory CAM's

In a static-memory system, CAM implementation would require separate logic at each memory word to compare that word with the reference word. If the results of the comparison of a particular word satisfies the comparison criteria, then a sense flag may be set at that memory location. Thus in order to implement a CAM search system alone would require a very large logical system. This large logic requirement is increased significantly by the logic required to operate on the word(s) which have had their sense flags set. Operation signals must be gated so that they reach only words with their sense flags on.

Another problem develops when it is desired to operate on only one word but many words satisfy the comparison criteria. For this case, logic must be provided to scan all of the sense flags and choose only one of them which is on and cause operation on that word alone.

Since such a large amount of logic is required to implement a CAM around a static memory, the cost of CAM systems have been prohibitive in most cases. The most promising approaches to developing a practical CAM system using static-memory elements involve systems using very small and very inexpensive logical and memory elements.

1.3.2 Circulating-Memory CAM's

The high cost of static CAM's can be avoided if use is made of a circulating memory system such as a magnetic drum, magnetic disc, or a delay line store. For these systems, the logic requirement would be greatly reduced because all words would not be compared to the reference word simultaneously. Thus a tradeoff is made between speed and cost. The circulating system is slower since comparisons are made in a serial fashion, but cost is reduced since one set of comparison and operate logic is shared by many stored words.

A circulating storage system can be searched in one circulation time of the memory.

1.4 Establishment of Design Goals

Since very few content-addressable memories have been constructued and since investigation of circulating CAM's have been

especially limited, it is desired to design and construct a high-performance, general-purpose circulating CAM. It is the purpose of this paper to report on the design and evaluation of this system.

1.4.1 Storage Media: Glass Delay Lines

Since speed is one of the principle advantages of a CAM, it is desirable to choose a circulating storage media which will operate at a very high rate. This eliminates magnetic drums and discs since their circulation times are measured in 10's of milliseconds. A random-access magnetic core memory system with a one microsecond cycle time could search on the order of 2000 words in ten milliseconds. So there would be small advantage to using a system with such a long circulation time.

A circulating memory system which is high-speed and has a short circulation time can be constructed using delay lines as the storage media. Glass delay lines are particularly attractive since they can be produced with a zero temperature coefficient (ZTC) of delay and can therefore be used conveniently in a synchronous storage system. ZTC glass delay lines are commercially available with delays of 100 microseconds and can be operated at up to 20 MHz bit rates. Even higher bit rates have been reported (3).

1.4.2 System Capabilities

The search criteria which the proposed system will have are as follows:

1. Equality
2. Greater
3. Less
4. Greatest
5. Least

The field of the word for which these comparisons are made is to be defined by a mask register. A bit position will be considered in the comparisons only if the mask register contains a one at that bit position.

The operations which the proposed system will be capable of performing are as follows:

1. Fetch
2. Store complete word.
3. Store only for bit positions which are one in the mask register.
4. Set or clear a tag bit for the word examined, for the next word, or for the previous word.
5. If an operation is to take place on only one word, read out its address.

The system will also have the usual random-access capabilities of storing or fetching a word at a given address.

1.4.3 Controlling Processor: NEBULA

There are no fixed requirements on the processor which would control a circulating CAM but the system which was available in this case is the NEBULA.

NEBULA is a general-purpose medium-speed serial computer which was designed and constructed at Oregon State University. It has 4096 34-bit words of memory stored in a glass delay line memory. There are 64 glass delay lines in the system which each store 64 words at a 22.735 MHz bit rate. The processor is synchronized with the memory and operates at a 350 KHz bit rate. The words in each delay line are interlaced so that a word is read out at exactly the processor bit rate. This organization fixes the processor word time at one memory circulation time which is 100 μsec. The processor is able to address any memory location with an effective-zero latency time.

The 34 bits in each word consist of a 32-bit data word (bits 1 to 32) preceeded by a spare bit (bit zero) and followed by a parity bit (bit 33). The bits are ordered least significant bit first. The number representation is two's complement and the floating-point format designates bit 32 as the sign of the fraction, bits 24 to 31 as the

exponent excess 128, and bits 1 to 23 as the fraction.

There is one instruction format which is basically that of a single address instruction format. An instruction consists of an 8-bit operation code, an 8-bit modifier, and a 16-bit address.

There are six flip-flop registers and seven memory locations which are used as index registers. Three of the flip-flop registers (U, V, and X) act as general arithmetic registers except for certain operations. The registers and their functions are listed.

U (34 bits) - Universal Accumulator

X (34 bits) - Extension of U

V (34 bits) - Operand Register

E (8 bits) - Input-Output Buffer

C (32 bits) - Program Control Counter (bits 17-32)

Flag Register (bits 1-16)

IR (32 bits) - Instruction Register

Operation Code, O (bits 1-8)

Modifier Code, M (bits 9-16)

Address Field, D (bits 17-32)

For a more detailed description of NEBULA, the reader is referred to the references (1, 2, 14).

For use with the CAM system, the following register assignment will be made:

U - Search Register

V - Mask Register

X - Address Read-Out Register

1.4.4 System Size: 2048 Words

Since NEBULA is to control the CAM and NEBULA has a 34-bit word length, it is most convenient to construct the CAM with a 34-bit word length.

Likewise, since NEBULA has a 100 microsecond word time, and since 100 microsecond, 20 MHz glass delay lines are readily available, the delay time for the delay lines used in the CAM is chosen as 100 microseconds. At 20 MHz, this allows 2000 bits to be stored in each delay line. Economic considerations limit the total number of delay lines to 35 which allows storage of 2000 words of 35 bits each. In order to allow a standard address of 11 bits, the capacity was chosen as 2048.

2. DATA ORGANIZATION

In the design of a circulating content-addressable memory, the method of organizing the stored data has a very strong influence on the hardware necessary to implement the system. In this chapter, three possible data organizations will be described and evaluated.

2.1 Interlaced-Vertical System

The interlaced-vertical system is identical to that used on the NEBULA computer for its random-access memory. In this organization, which stores 65 words of 35 bits each in every delay loop, an observer at an output read amplifier would first observe 65 first bits, then the 65 second bits, and so on. The bits for any given word are thus spaced 65 bits apart in the memory. Figure 2.1 gives a schematic representation of such a memory with word #2 of delay loop #1 addressed.

Figure 2.2 illustrates the timing system used for this system. The circulation time is divided into 35 macrobit times, T_0 through T_s, which are in turn divided into 65 microbit times, t_0 through t_x. The random-access addressing scheme requires that T_s and all of the times t_x be reserved for timing purposes so that each delay loop can in fact store only 64 words of 34 bits each. Thus, 32 delay loops will provide the desired 2048 word memory capacity.

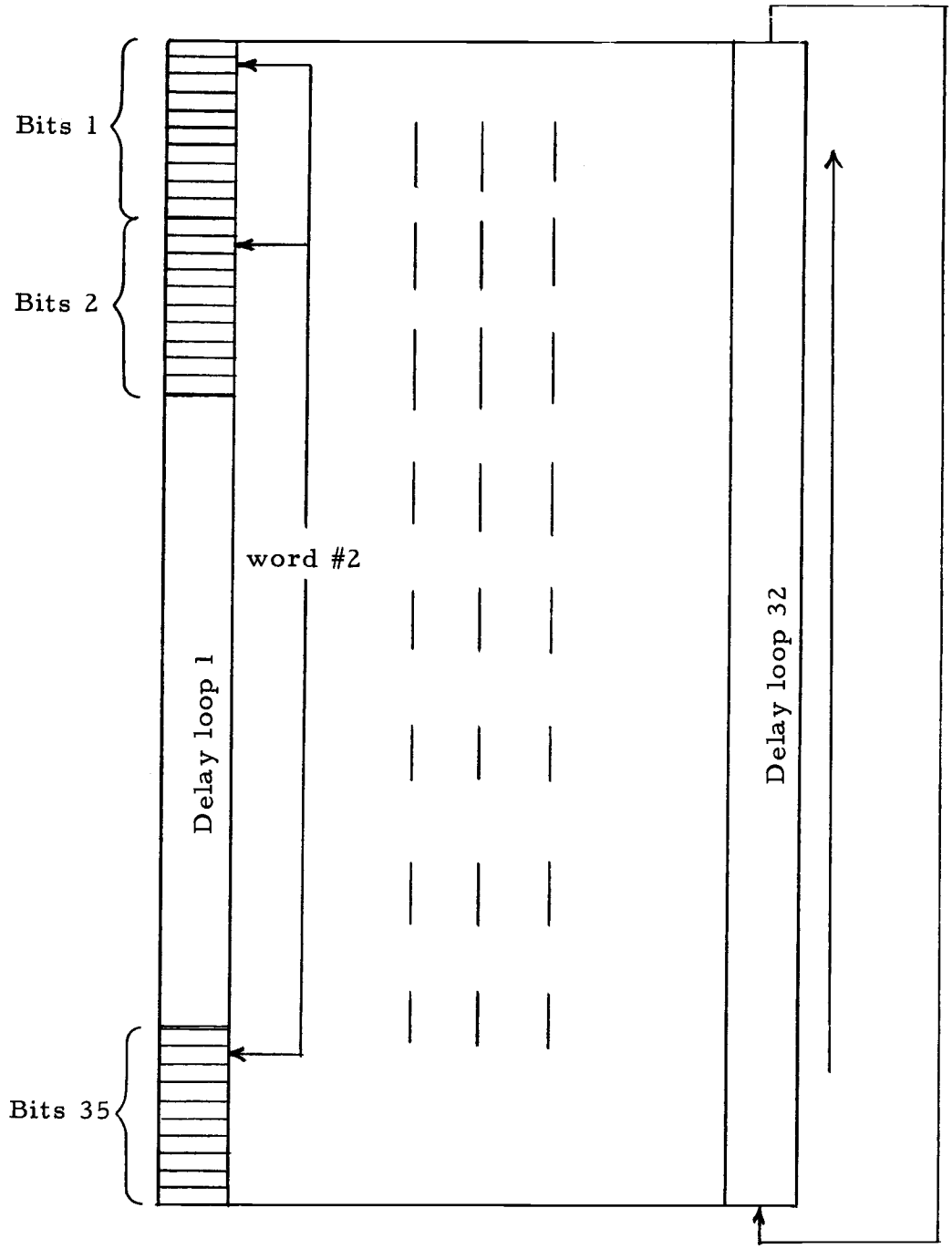


Figure 2.1. Interlaced-vertical system schematic.

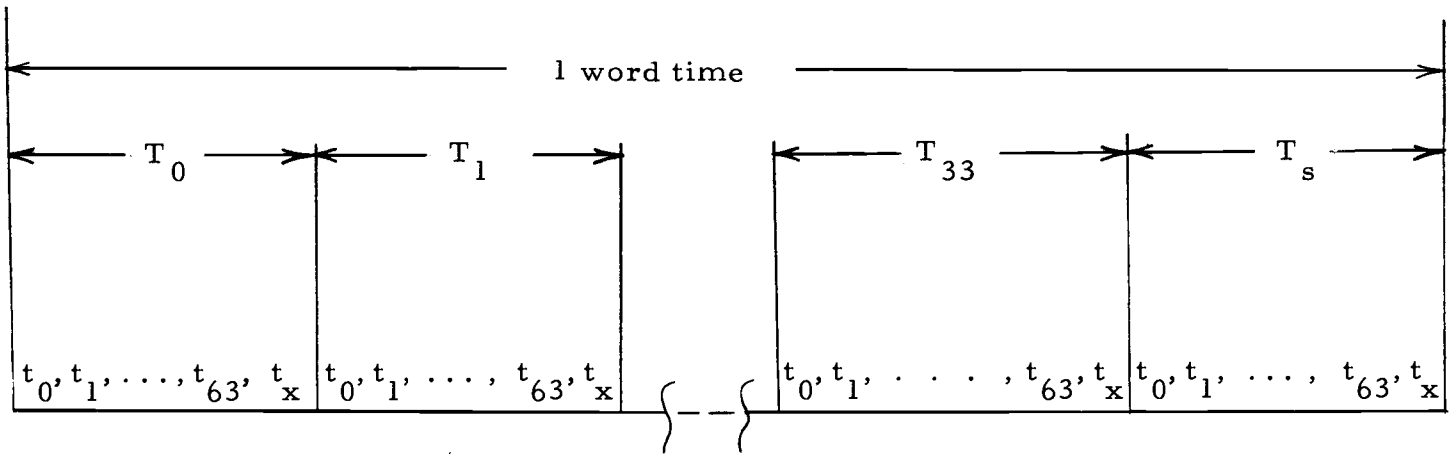


Figure 2.2. Interlaced-vertical timing.

2.1.1 Random-Access Addressing

Random-access addressing is accomplished from the 11-bit address by means of a logic decode of 5 bits to select the delay line and then equality of the low-order 6 bits with the microbit counter to address the proper word within the selected delay line. See Figure 2.3.

Figure 2.3 also shows how an address may be generated when an operation is to be performed on only one word. A signal from the delay line containing the word will activate the address coder to generate 5 bits of the address code. When the word in question appears at the delay line output, the microbit counter's state can be gated out to form the remaining 6 bits of the address.

2.1.2 Search Algorithms

Comparisons are made between the search register (U) and memory on a serial-by-bit basis. During macrobit time T_0, U_0 is compared with all bits #0 in the memory, during T_1, U_1 is compared with all bits #1, and so on. And during T_{33}, all bits #33 are compared with U_{33} to complete the comparison.

Notice that one circulation time is required before a complete comparison can be made on any word in memory. Since comparison information must be maintained for each word during the comparison

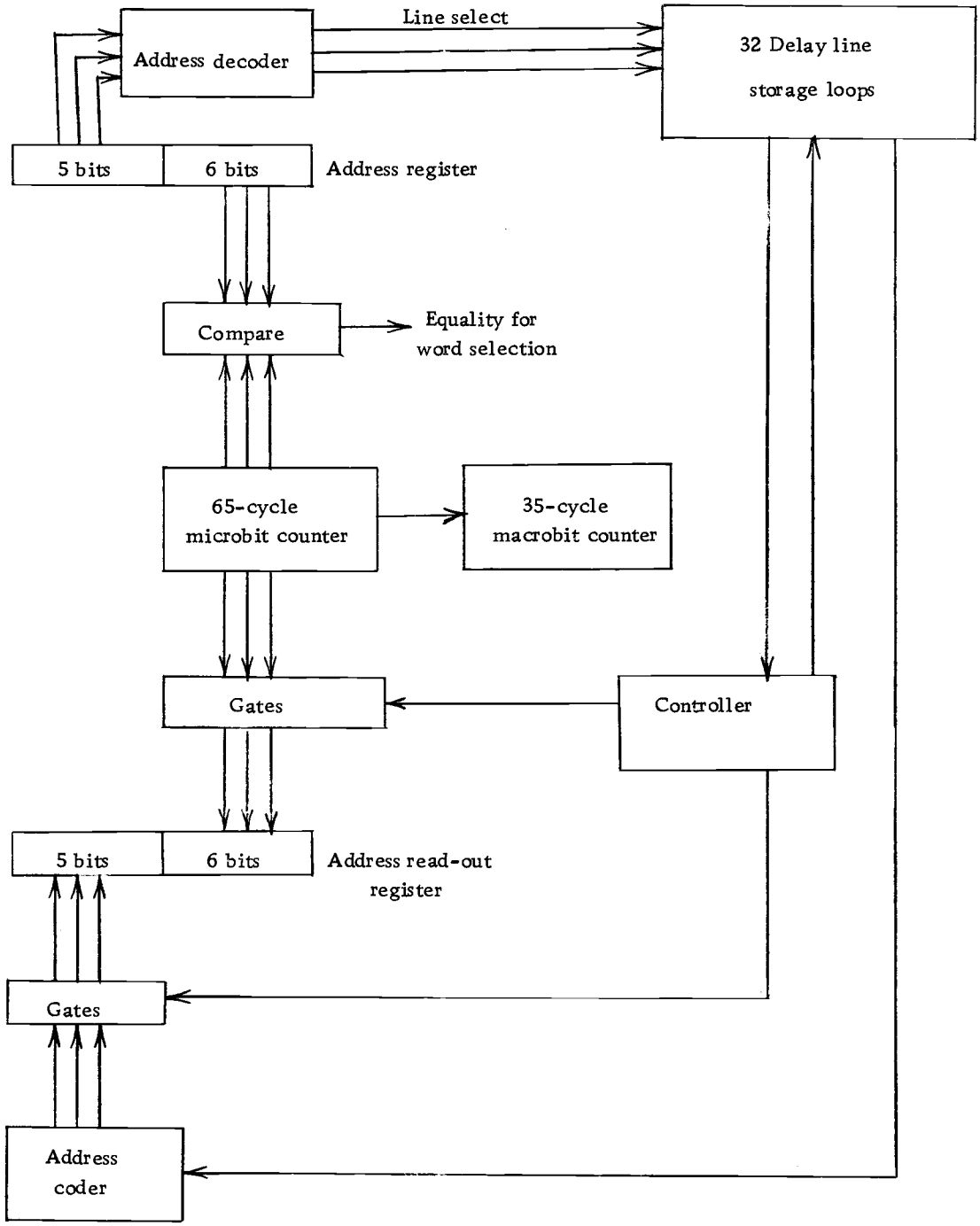


Figure 2.3. Interlaced-vertical random-access system.

circulation time, a sense register must be available with one bit position for each word in memory. One such sense register is required for an equality search and two sense registers are required for magnitude searches.

Figure 2.4 illustrates the i^{th} storage loop (i ranges 1 through 32) and the sense registers, a_i and b_i which are shown as circulating registers constructed from delay lines of 65 bits length. The sense flip-flop G_i is used to indicate when at least one bit is on in the sense register a_i. The sense registers a_i and b_i are synchronized with the storage delay loop so that when a bit from word #j appears at R_i, the loop output, the sense register bits corresponding to word #j will appear at R_{ai} and R_{bi}. During this bit time, R_i will be compared with U and if necessary, the sense loops can be interrupted by signals S_{ai} or S_{bi} to store D_{ai} or D_{bi} into the sense registers.

Bits in sense register a_i can be set for words equal to the search register (U) by the following procedure. During T_0, a sense bit is set if $U_0 = R_i$. For T_1 through T_{33}, the sense bit is cleared if U_j does not equal R_i during T_j. At the end of the T_{33} the search is complete.

If the words are stored in memory so that the most significant bit appears first, magnitude searches can easily be executed.

Bits in sense register a_i can be set for words greater than

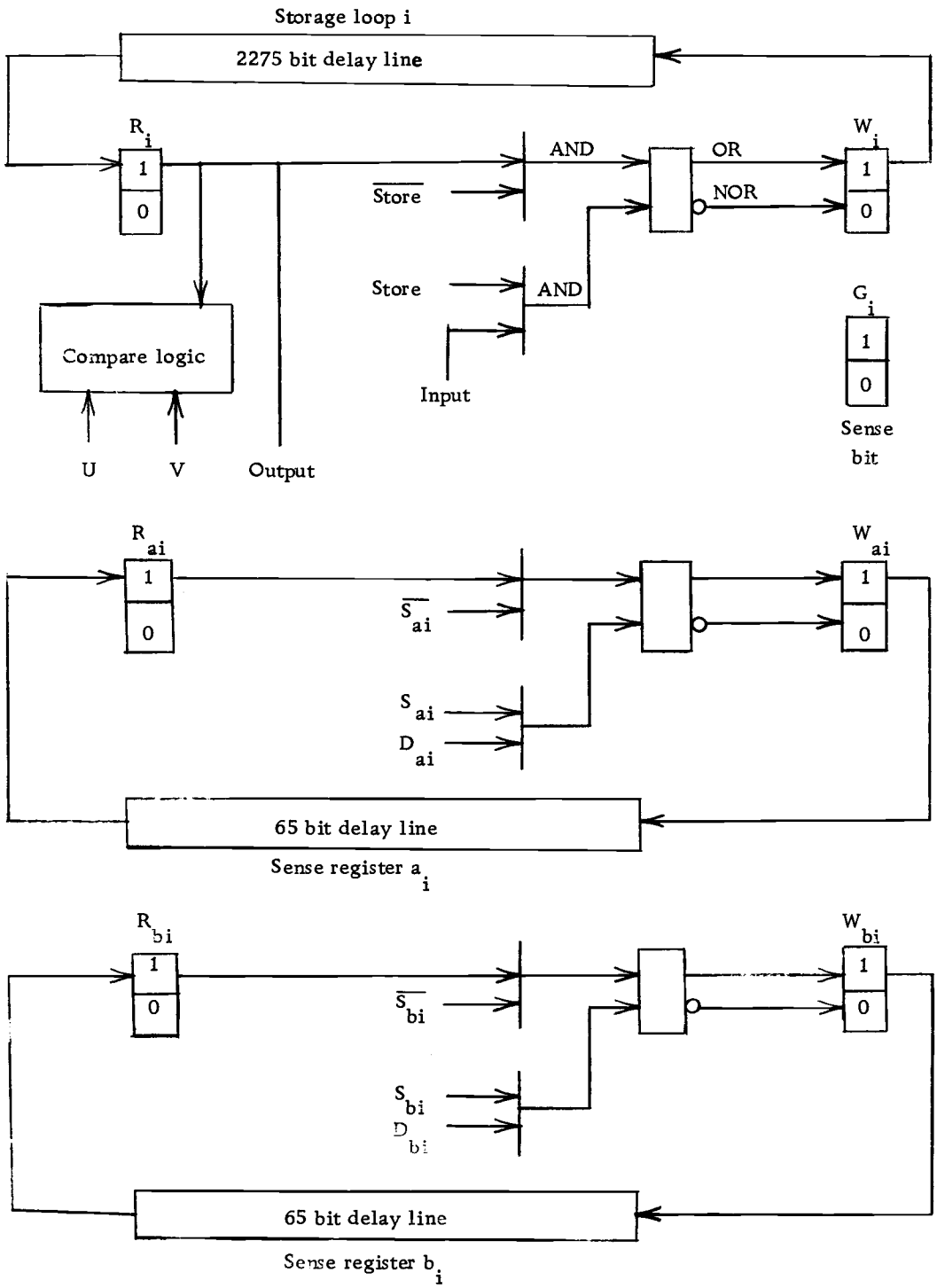


Figure 2.4. Interlaced-vertical storage loop and sense registers.

the search register (U) by beginning with all bits of a_i and b_i being set. A sense bit in b_i is cleared when R_i does not equal U. If at this time R_i is less than U, the sense bit in a_i is cleared. After a sense bit in b_i is cleared, the corresponding bit in a_i cannot be cleared. This leaves one bits in a_i only for words which are greater than U. The search for words less than U is equivalent.

It is also possible to conduct a search for the maximum word in memory with use being made of the flip-flops G_i and a central flip-flop, P. The search is started with P and all bits in a_i and b_i being one. During T_0, bits in a_i will be cleared if R_i is zero. If at t_x of T_0 all bits in a_i have been cleared, G_i will be turned off. And if all the G_i's are zero at t_x, then P will be turned off. If P is off, this indicates that all a_i sense bits are off and that no word is maximum. This of course is an invalid conclusion so b_i is used to store the latest "legitimate" search information. If P is on, this indicates that all words were not eliminated and that a_i contains valid search information. Then b_i will copy the information from a_i. If this procedure is repeated during T_0 through T_{33}, a_i will indicate the maximum word(s) in memory.

A similar procedure can be used to locate the ~~minimum~~ smallest word(s) in the memory.

2.1.3 Operations

As previously stated, one circulation time has been required in order to isolate a word as satisfying a given search criteria. After a word has been established as satisfying the search criteria, another circulation time is required to operate on that word. The search-operate time for the interlaced-vertical system is then two circulation times.

The operations of storing, fetching, and tagging are straightforward in implementation. Difficulty arises however, if many words satisfy the search criteria but an operation is to be performed on only one location. Logic must be provided to choose one of the delay loops which has one bits in its sense register. That is to say, if a search leaves several G_i flip-flops in the one state, it is necessary to choose the "first" delay loop which has G_i one. When one delay loop is chosen, then it is necessary to choose the "first" word in this delay loop which has a one bit in the sense register a_i and to operate on that word.

This problem of several words sensed when only one is desired is the "multiple-match" problem.

2.2 Contiguous-Vertical System

The organization of information in the contiguous-vertical

system is shown in Figure 2.5. This organization has some similarity to the interlaced-vertical system but now, an observer at the output of the first delay line observes all of word #1, then all of word #2, and so on. The bits of a given word still appear serially, but they are now adjacent to each other rather than interlaced with bits from other words.

By dividing one delay-loop circulation time as shown in Figure 2.6, it is possible to store 64 words in word times T_1 through T_{64}. It is convenient to leave an unavailable bit time, t_x, at the end of each word time and a spare word time, T_s, at the end of the circulation time. T_s and the times t_x are used for decisions at the end of each word time and at the end of the circulation time.

2.2.1 Random-Access Addressing

The random-access addressing for the contiguous-vertical system is very similar to that used for the interlaced-vertical system. Figure 2.7 shows how a word is addressed by using five bits of the address to select the delay loop, and a comparison of six bits of the address with the word counter to select the word in the delay loop.

When an operation is to be performed on only one word, its address can be generated by coding five bits of the address from the delay loop location, and by gating the contents of the word counter when the desired word appears at the delay loop output.

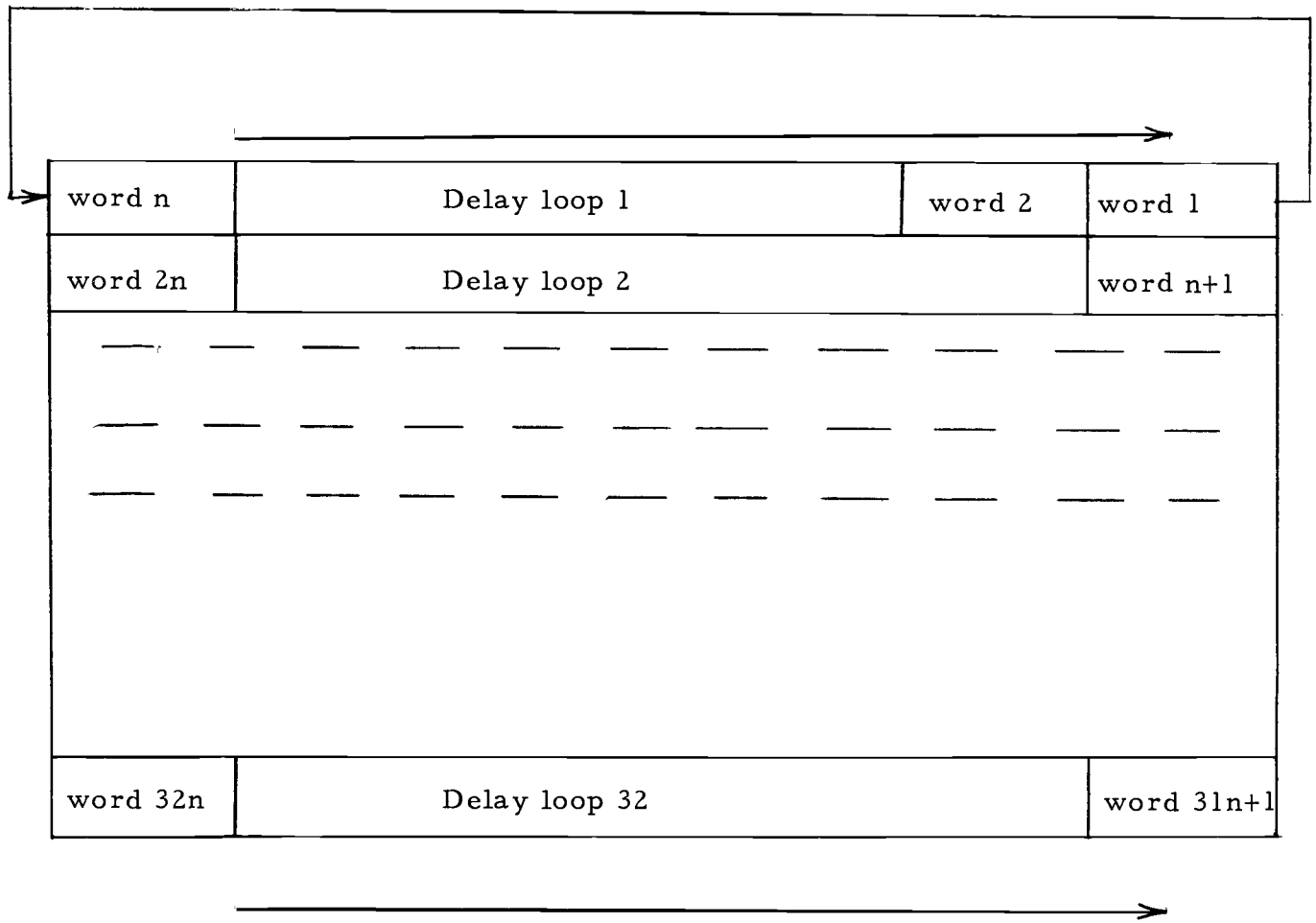


Figure 2.5. Contiguous-vertical system schematic.

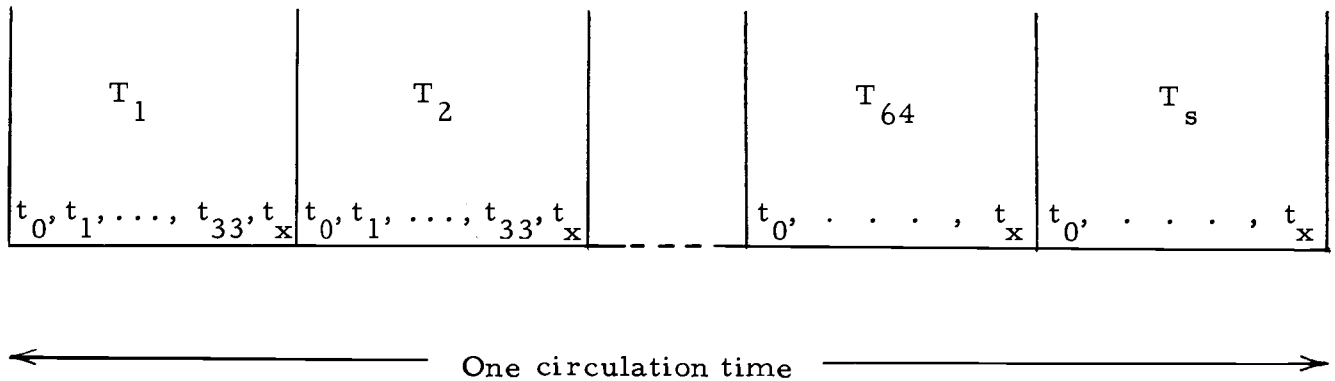


Figure 2.6. Contiguous-vertical system timing.

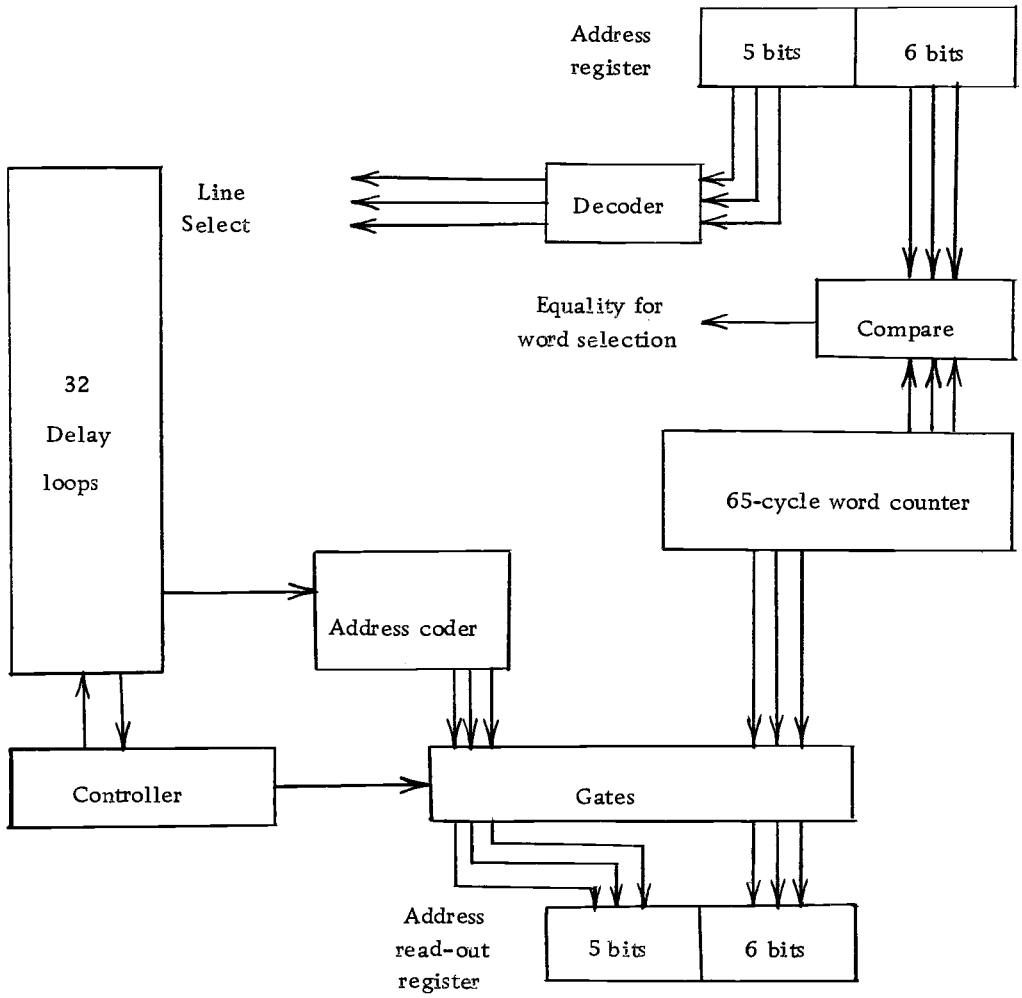


Figure 2.7. Contiguous-vertical random-access system.

2.2.2 Search Algorithms

In the continguous-vertical system, comparisons between the search register and the memory are made on a serial-by-bit basis so that the algorithms are quite similar to those used for the interlaced-vertical system. However, now all of one word will appear at a delay line output, R_i, before any information from the next words will appear. This allows word #j to be compared with the search register during T_j, and at t_x of T_j, the control logic can decide whether to operate on word #j. If word #j is to be operated on, the operation can take place during T_{j+1} as word #j appears at the output of the word delay, D_i (see Figure 2.8).

Therefore, within one delay loop, a word is compared with the search register and the decision to operate is made before the next word appears. This relieves the requirement for the sense registers used in the interlaced-vertical system. Now each delay loop needs only three flip-flops, G_i, H_i, and O_i (see Figure 2.8). G_i and H_i are used as sense bits during a comparison of a word at R_i with the search register and O_i is used to indicate when an operation is to be performed on the word appearing at D_i.

The search algorithms for words equal to, greater than, or less than the search register are the same as those used in the interlaced-vertical system with the difference being that G_i and H_i

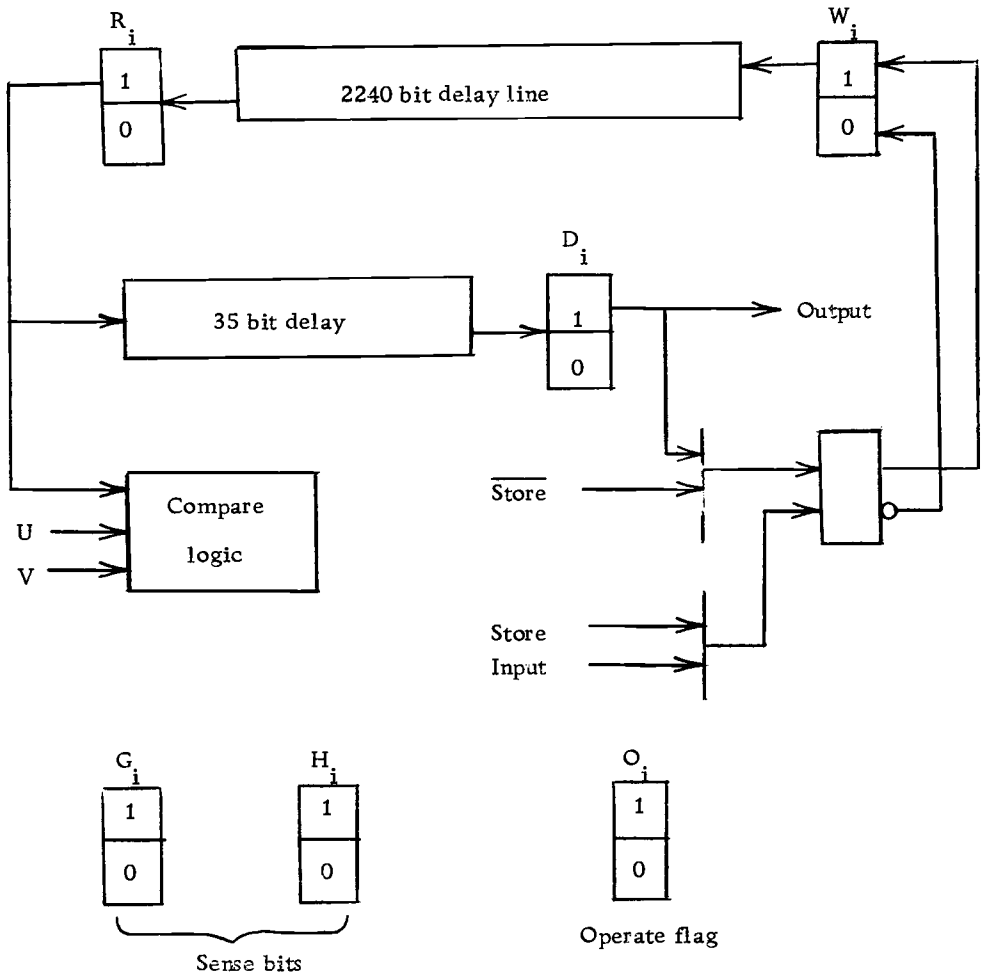


Figure 2.8. Contiguous-vertical storage loop and sense bits.

are used as sense bits instead of bits from a_i and b_i as in the interlaced-vertical system.

To illustrate, consider a search for words greater than the search register. At the beginning of word time T_j, G_i and H_i are both one. The most significant bit of word #j is compared to the most significant bit of the search register during t_0. The next most significant bit is compared during t_1, and so on. G_i and H_i are left unchanged until a bit position is reached where R_i does not equal U. At this time, H_i is cleared and G_i is cleared if R_i is less than U at this bit position. After H_i is cleared, G_i is left unchanged. At time t_x, word #j will be greater than U if G_i is one.

Searches for maximum and minimum are somewhat different than the algorithms used in the interlaced-vertical structure.

For the search for largest, the search register begins holding the smallest possible word. During T_1, a search is made for the largest R_i which is also greater than U. This word is found by looking for words greater than U in the manner described above. However, as soon as one H_i is turned off but G_i is left on (indicating that a word greater than U has been found at R_i) all other G's are cleared. At t_x, O_i is set to one and during T_2, D_i is loaded into U and D_i is used as the comparison word during T_2. If this sequence is repeated, U will contain the largest word

in memory at T_s.

The search for smallest is equivalent.

2.2.3 Operations

The contiguous-vertical system requires one circulation time of the delay loops in order to search all memory locations. However, since operations can be taking place as soon as a "match" is found, the search-operate time for this system is also one circulation time.

Storing, fetching and tagging functions are easily implemented in this system but again it is necessary to provide logic to choose one delay loop when several delay loops have words which satisfy the search criteria.

2.3 Horizontal System

The horizontal system organization is illustrated in Figure 2.9. In this system, one delay loop is required for each bit of the words stored. A 34-bit word length would require 34 delay loops and the 34 outputs, R_0 through R_{33}, display words parallel-by-bit and serial-by-word.

The timing structure for this system consists simply of 2048 time divisions t_0 through t_{2047}. During t_i, the read register (R) contains word #i, during t_{i+1}, R contains word #i+1, and so on. After one circulation time the entire memory contents have

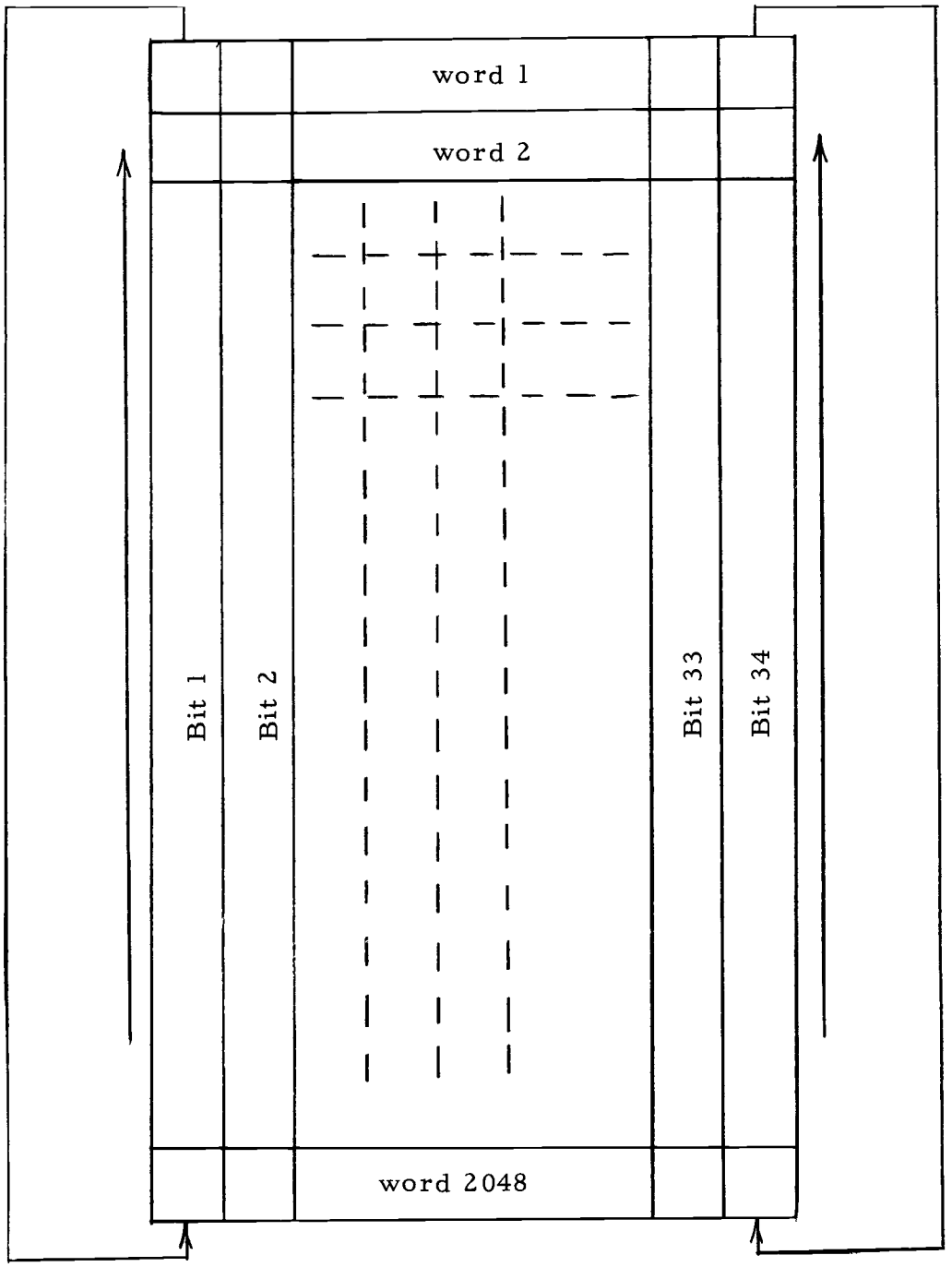


Figure 2.9. Horizontal system schematic.

passed through R.

2.3.1 Random-Access Addressing

Random-access addressing can be achieved with the horizontal system by synchronizing a 2048 cycle counter with the delay loops. This will associate an 11-bit address with each word in the system.

As shown in Figure 2.10, any word can then be addressed by comparing the address register with the address counter. When these two are equal, the word addressed is in the read register (R).

When an operation is to be performed on only one word, its address can be generated simply by gating the contents of the address counter into the address read-out register.

2.3.2 Search Algorithms

In the horizontal system (see Figure 2.11) the search algorithms are very simple since comparisons are made in parallel between the search register and the read register. The comparison logic generates an equality signal (E) and a greater-than signal (G).

$$\overline{E} = \sum_{i=0}^{33} (R_i \overline{U}_i + \overline{R}_i U_i) V_i$$

$$G = V_{33} R_{33} \overline{U_{33}} + \sum_{i=0}^{32} V_i R_i \overline{U}_i \left[\prod_{j=i+1}^{33} \overline{(V_j \overline{R}_j U_j)} \right]$$

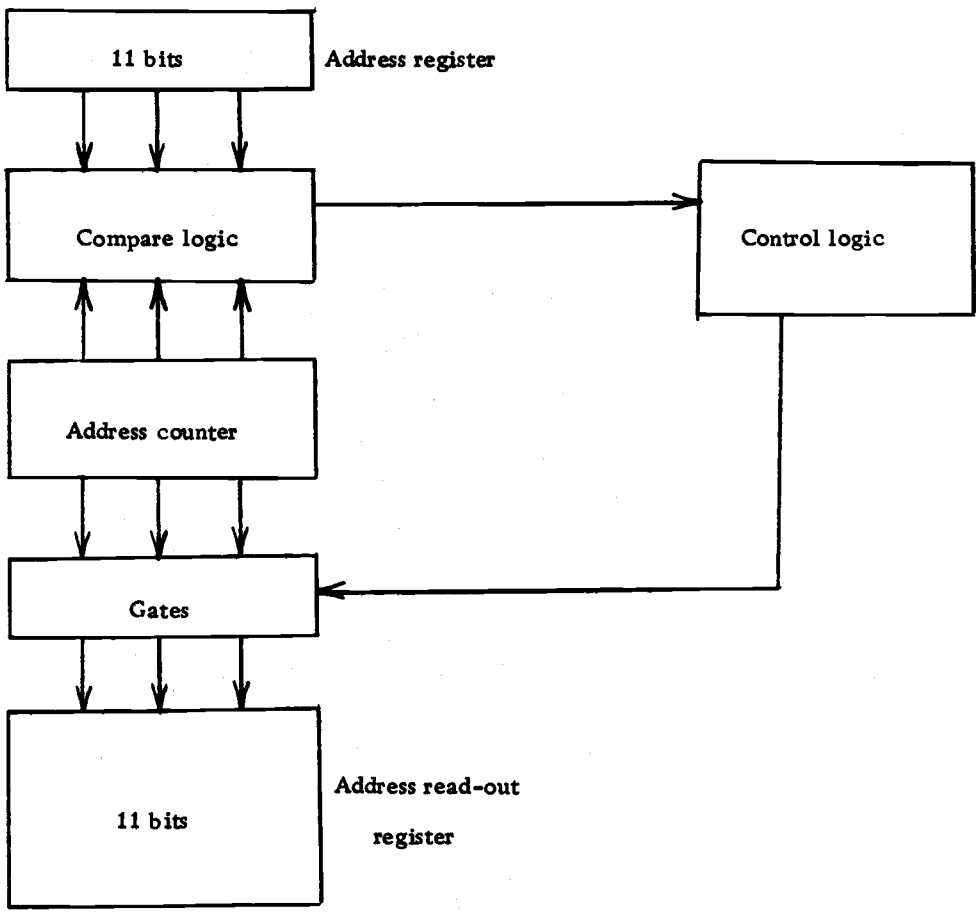


Figure 2.10. Horizontal random-access system.

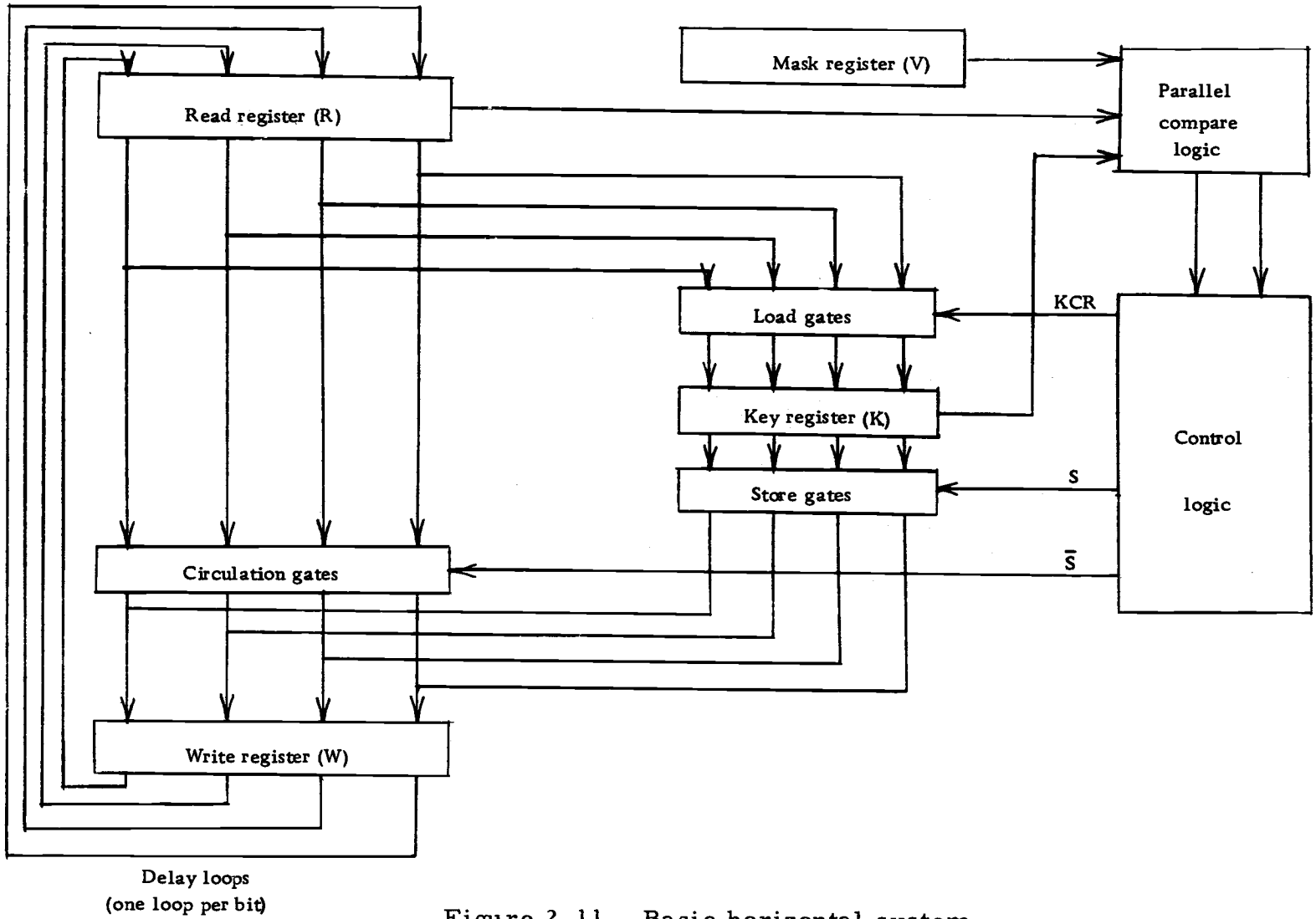


Figure 2.11. Basic horizontal system.

Searches are conducted by simply considering the value of E and G for each word that appears at R. Sense registers or bits are not required since as soon as a word is found that satisfies the search criteria, it can be operated on immediately.

For a search for maximum, the smallest word possible is placed in the search register (U). Then as words pass through R, they are compared with U and if R is greater than U, that word is immediately gated into U. If this process is repeated until all words have passed through R, then the largest word in memory will be located in U.

So in the horizontal system, the search algorithms are not dependent on timing and control but rely instead on the logical functions E and G.

2.3.3 Operations

The horizontal system requires one circulation time to search all of memory and since operations take place as soon as a word is found which matches the search criteria, the search-operate time is also one circulation time.

Storing, fetching, and tagging operations are easily implemented and it should be noted that there is no problem to operate on only one word. A search can simply be terminated after the first word is found which satisfies the search criteria.

2.4 Comparative Evaluation

Each of the three data organizations, interlaced-vertical, contiguous-vertical, and horizontal could be used as a data structure for a circulating content-addressable memory. However, the peculiarities of each system demand the use of different search algorithms and in turn, different logical control systems.

Some of the salient features of each system are shown in Figure 2.12 for comparison.

The search capabilities of all three systems are the same, but the search-operate times show the interlaced-vertical system to be twice as slow as either the contiguous-vertical or the horizontal system.

Comparisons are made serial-by-bit in the two vertical systems and parallel-by-bit in the horizontal system. A serial-by-bit comparison requires much less logic than a parallel comparison but this logic savings is offset by the fact that 32 sets of serial-by-bit logic are required for the vertical systems.

The interlaced-vertical system requires sense registers to provide sense bits for each word in the memory. This leads to the requirement of 64 delays of length 65 bit-times to be used as sense registers. Logic must be provided to control each of these sense registers and to provide solution to the multiple-match problem.

	Interlaced vertical	Contiguous vertical	Horizontal
Search-operate times	2 circulation times	1 circulation time	1 circulation time
Comparisons	Serial-by-bit parallel-by-word	Serial-by-bit mixed serial and parallel-by-word	Parallel-by-bit serial-by-word
Sense register requirements	bits for each word	bits for each loop	bits for system only
Hardware	32 delay loops 64 65-bit delays	32 delay loops 32 35-bit delays	35 delay loops

Figure 2.12. Comparison of data organizations.

The contiguous-vertical system requires sense bits only for each delay loop and therefore requires significantly less control logic than the interlaced-vertical system. However, the multiple-match problem still exists although on a smaller scale since there are only 32 sense bits instead of 2048 for the interlaced-vertical system. Also, the continguous-vertical system requires hardware to produce a 35 bit-time delay in each delay loop.

The horizontal system has the simplest control logic of the three systems since it need not control more than one sense bit and it is not faced with the multiple-match problem.

In choosing the data organization to be used in the final system, the interlaced-vertical system was eliminated because it is the slowest and also requires the most hardware. The horizontal system and the continguous-vertical systems are comparable in performance, but the horizontal system was chosen for detailed design and construction because of its simplicity. It requires less control logic and does not need the 35-bit delays that the continguous-vertical system requires.

It should be noticed, however, that if a content-addressable memory were to be constructed from one or a few delay loops, the horizontal system would be impossible to use and the continguous-vertical system would be recommended.

3. THE NEBULA CAM SYSTEM

The horizontal data organization has been established as the best performer and the most economical for implementation of the 2048 word memory when approximately 2000 bits can be stored in each delay loop. This chapter will describe in detail the horizontally organized, content-addressable memory system which has been constructed and installed as an additional memory for the NEBULA computer.

The design philosophy has been to develop a memory system with a maximum of performance and flexibility but without a major sacrifice in economy. Performance and economy are quite often conflicting goals so that a balance must be reached between the two. Features were not added to this memory unless they provided a significant increase in performance or flexibility with a minimum cost.

Figure 3.1 illustrates the memory system's capacity and the functional assignments of the registers. The memory will store 2048 words of 35 bits each. Bits #0 through #33 are data bits with bit #33 being designated as the tag bit. The 35th bit in each word is not accessible as data and is called the use bit. The use bit is provided to indicate when useful information is stored at a location.

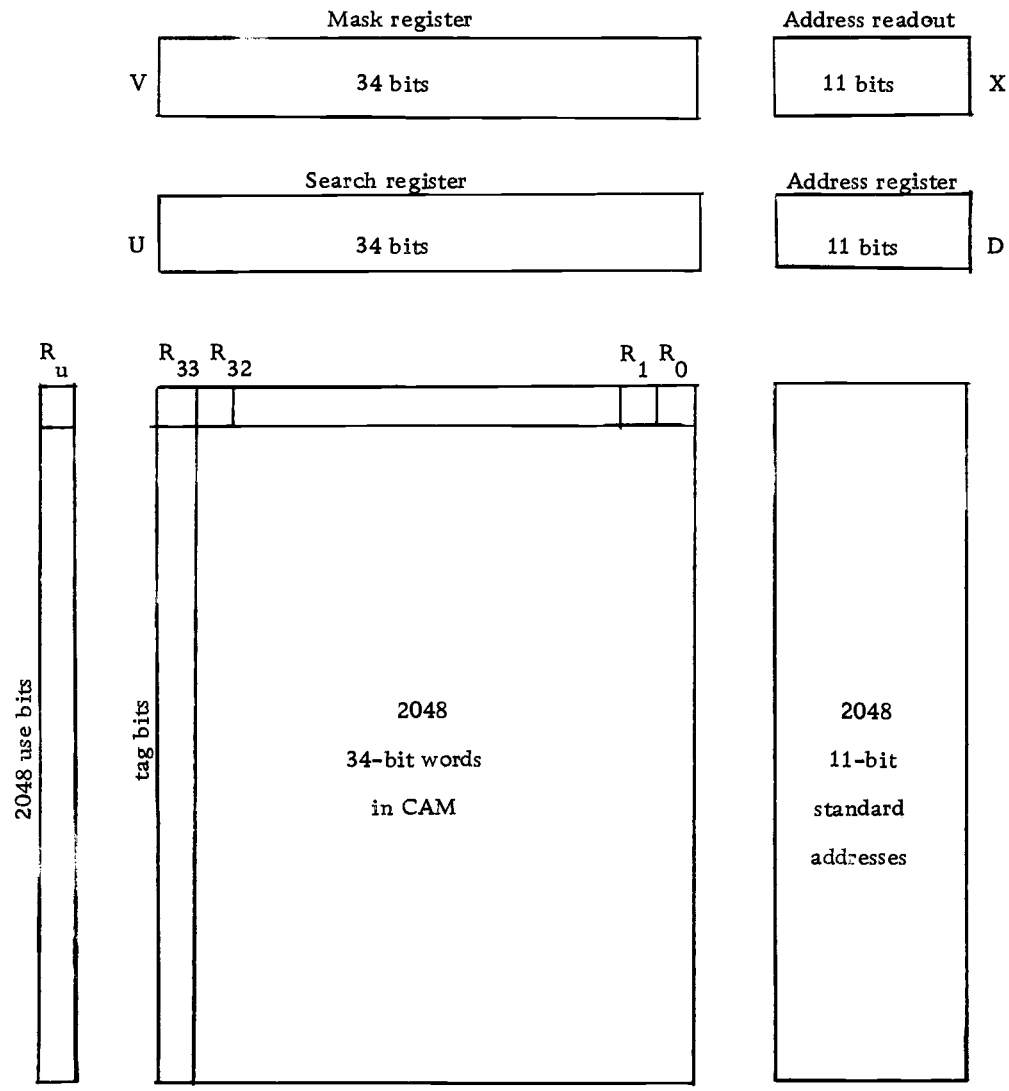


Figure 3.1. NEBULA CAM system layout.

3.1 Coding System and Command Set

The instruction word format used on NEBULA breaks the 32 bit instruction into an 8-bit operation code (stored in the O register), an 8-bit modifier (stored in the M register), and a 16-bit address (stored in the D register). Since CAM instructions do not designate an address, CAM instructions use a single operation code and make use of the 16-bit address field to specify the search criteria desired and the operations to be performed. Each bit in the address field is used to designate a particular function as listed in Figure 3.2. The modifier field is cleared when a CAM instruction is fetched.

3.1.1 CAM Instructions

Content-Addressed Memory Commands:	Mneumonic Code	Hexadecimal Code
Execution time: 0 or 100 microsec.	CAM	04

Search Criteria:

Five bits (D_1 through D_5) of the CAM instruction's address field are used to establish the criteria for the search. A memory word will be operated on if:

1. The word's use bit = USE

AND 2. if TAG = 1, the word's tag bit is one,

AND 3. if GTR = 1, the word is greater than U,

OR, if EQL = 1, the word is equal to U,

	Function	Mneumonic code	Bit #
Operations	Store U Associatively	(SUA)	D_{16}
	Store U Selectively	(SUS)	D_{15}
	Load U Associatively	(LUA)	D_{14}
	Operate on ALL Words	(ALL)	D_{13}
	Clear Previous Tag	(CPT)	D_{12}
	Set Previous Tag	(SPT)	D_{11}
	Clear Same Tag	(CST)	D_{10}
	Set Same Tag	(SST)	D_{9}
	Clear Next Tag	(CNT)	D_{8}
	Set Next Tag	(SNT)	D_{7}
	Change Use Bit	(CHU)	D_{6}
Search criteria	Use Bit	(USE)	D_{5}
	Tag Bit	(TAG)	D_{4}
	Greater	(GTR)	D_{3}
	Equal	(EQL)	D_{2}
	Less	(LES)	D_{1}

Figure 3.2. CAM instruction format.

OR, if LES = 1, the word is less than U,

OR, (GTR)(EQL)(LES) = 1.

The comparisons between U and memory words are made using only bit positions of the word where the V register has ones.

If a CAM instruction is executed and no word is found which satisfies the search criteria, then a flag, C_9, is set in the NEBULA processor.

Operations:

Each of the remaining 11 bits (D_6 through D_{16}) specifies a separate operation.

SUA(D_{16}) Store U at the locations which satisfy the search criteria.

SUS(D_{15}) Store U at the locations which satisfy the search criteria. Bit U_i is stored only if $V_i = 1$.

LUA(D_{14}) Load U with contents of the location which satisfies the search criteria.

ALL(D_{13}) For ALL = 0, the search is terminated when one word is found that satisfies the search criteria. The address of this word will be stored in the left half of the X register. For ALL = 1, the specified operations take place on all words which satisfy the search criteria.

CPT(D_{12}) Clears the tag (bit #33) on the previous word for locations which satisfy the search criteria. GTR and LES are

considered zero.

SPT(D_{11}) Sets the tag on the previous word for locations which satisfy the search criteria. GTR and LES are considered zero.

CST(D_{10}) Clears the tag on words which satisfy the search criteria.

SST(D_9) Sets the tag on words which satisfy the search criteria.

CNT(D_8) Clears the tag on the next word for locations which satisfy the search criteria.

SNT(D_7) Sets the tag on the next word for locations which satisfy the search criteria.

CHU(D_6) Changes the state of the use bit on words which satisfy the search criteria.

Each of these operations is independent but the effects of certain combinations of them must be noted.

1. If the tag is being both set and cleared simultaneously, the clearing will be dominant. For example, if CST and SST are both one, then only CST will operate. Also, if a word is being tagged and a zero is being stored in bit #33, bit #33 will be zeroed.
2. If SUA and SUS are both one, then only SUA will operate.
3. If SUA or SUS is one and LUA is one, then an exchange will take place between the memory location

satisfying the search criteria and the U register.

4. If LUA and ALL are both one, all tagging, storing and use-bit operations are inhibited and EQL is ignored. If $(LUA)(ALL)(GTR)(\overline{LES}) = 1$, then the greatest word satisfying the search criteria is loaded into the U register provided this word is greater than the initial value of U. And if $(LUA)(ALL)(LES)(\overline{GTR}) = 1$, then the least word satisfying the search criteria is loaded into the U register provided this word is less than the initial value of U. These two combinations provide commands to fetch the greatest and fetch the least words from memory.

3.1.2 Random-Access Instructions

The memory is designed so standard random-access addressing is also possible. Three instructions using the standard NEBULA instruction format control the random-access functions. These instructions may undergo the usual address modification.

Random-Access Fetch:	Mneumonic Code	Hexadecimal Code
$C(U) := C(A)_{CAM}$	RAF	01

The U register is loaded from the CAM with the contents of the location specified by the effective address. If that location has use-bit zero, the processor flag, C_9, is set.

Execution time: 100 microseconds.

Random-Access Store and Clear

Use Bit:	Mneumonic Code	Hexadecimal Code
$C(A)_{CAM} := C(U)$	RASC	02

The contents of the U register is stored in the CAM at the location specified by the effective address. Also, the use bit is cleared at that location.

Execution time: 100 microseconds.

Random-Access Store and Set

Use Bit:	Mneumonic Code	Hexadecimal Code
$C(A)_{CAM} := C(U)$	RASS	03

The contents of the U register is stored in the CAM at the location specified by the effective address. Also, the use bit is set at that location.

Execution time: 100 microseconds.

3.2 CAM Interface with NEBULA

The normal sequence of operation for NEBULA calls for one word time (100 microsecond) in state F_0 to fetch an instruction. Then the state will change to F_3 for execution of the instruction. Intermediate states F_1 or F_2 will be executed if address

modifications are desired. After completion of the instruction, NEBULA will return to F_0 and fetch the next instruction.

For CAM instructions, state F_1 and F_2 are not used since all address modifications are prevented. It is also possible to avoid the execution state F_3 for CAM instructions by executing them during the F_0 which fetches the next instruction, In order to accomplish this effective-zero execution time, there must be some means of storing the CAM instruction while the next instruction is being fetched. This is done by storing the CAM instruction address field in the CAM instruction register (DI), shown in Figure 3. 3, while the CAM instruction is being executed and the next instruction is being fetched. The CAM instructions require 100 microseconds for execution so that CAM instruction execution can be completely concurrent with the following instruction fetch.

However, difficulty occurs when an attempt is made to execute two CAM instructions successively. When the second CAM instruction is fetched it cannot be stored in DI since DI holds the previous CAM instruction which is being executed. In order to take care of this situation, the second CAM instruction will cause NEBULA to go to state F_3 for one word time before executing the second CAM instruction. During F_3, DI is loaded with the address field of the second CAM instruction. CAM instructions also always spend one word time in F_3 when being one-stepped. This facilitates

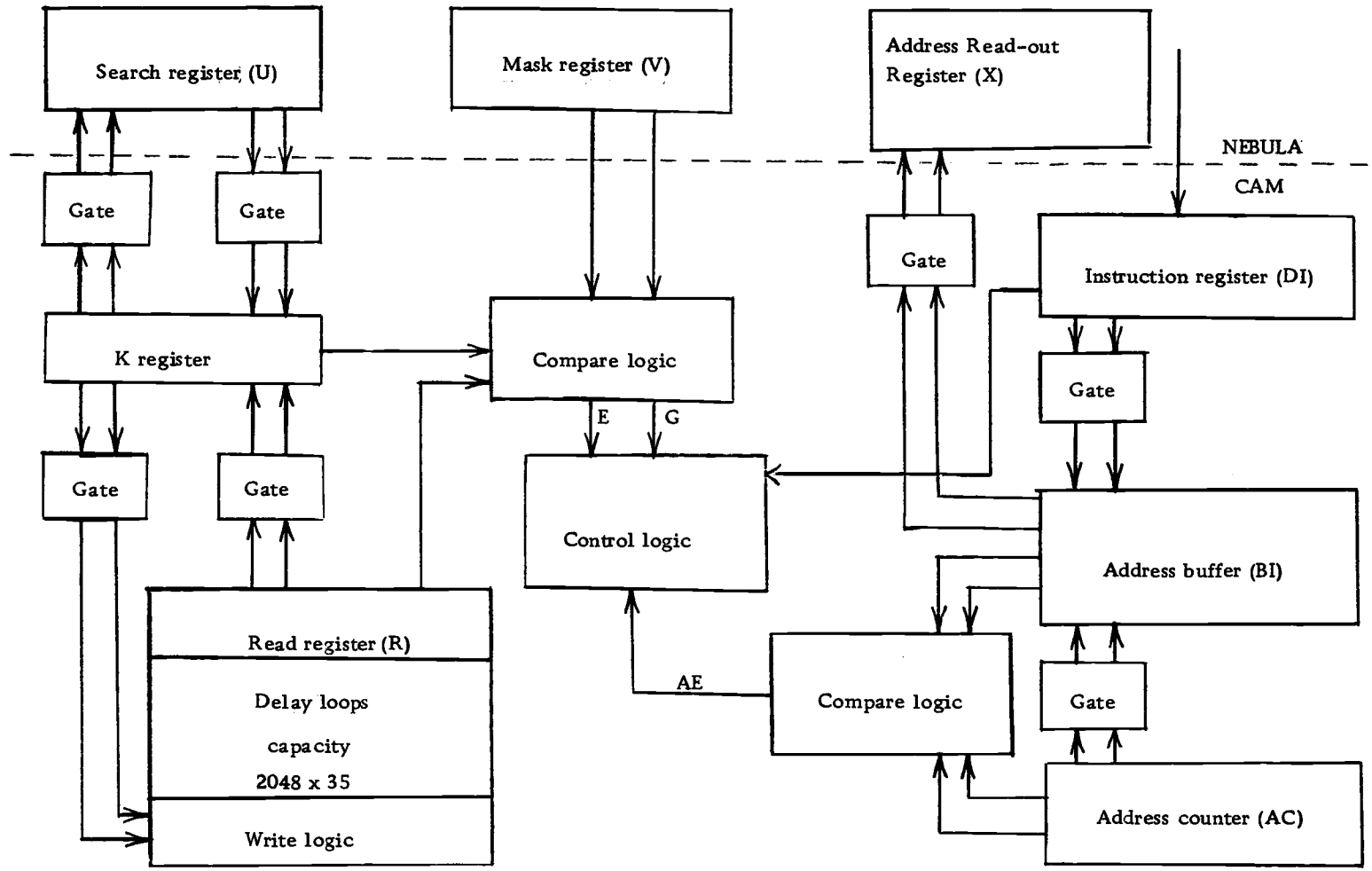


Figure 3. 3. NEBULA CAM system.

observation of the CAM instruction execution.

The execution of a CAM instruction involves first transferring the contents of the search register (U) to a high-speed search register (K). This transfer is required since NEBULA's U register is not fast enough to operate at the 20 MHz rate of the memory.

Once the search information is in the K register, comparisons are made between K and the delay loop output register (R). The control logic will consider the comparison signals E and G, and produce operations whenever the comparisons satisfy the search criteria of the instruction. If necessary, the contents of K can be transferred back to U at the end of the instruction execution.

The random-access instructions require very little change in the sequence used for the CAM instruction. However, since the random-access instructions can be address modified, the normal NEBULA F-state sequence is left uninterrupted and these commands will always spend one word time in F_3. During F_3, DI is loaded with the instruction address as modified, then DI is transferred to the address buffer (BI) and DI is cleared to prevent actuation of any CAM command operations. The address buffer is compared with the address counter (Figure 3.3) in order to effect the random-access addressing.

The CAM uses 100.000 microsecond delay lines as does the random-access memory which controls the timing of NEBULA's

processor. However, the circulation times, including electronics, of the CAM memory and the random-access memory are slightly different and the clock frequencies are 20.446 MHz and 22.735 MHz respectively. The two systems are not synchronized and therefore, when the NEBULA processor initiates execution of a CAM command, the starting address in the CAM is random.

3.3 Description of CAM System Logic

3.3.1 NEBULA Processor Logic

The processor logic is written for NEBULA processor circuitry which uses clocked R-S flip-flops and is written in the following form:

$_1X_3$ is the clocked set function for flip-flop X_3.

$_0X_3$ is the clocked reset function for X_3.

$X_3 \leftarrow X_2$ means X_3 copies X_2.

RAF, RASS, RASC, and CAM are instruction decodes from the O register. Figure 3.4 illustrates NEBULA processor timing.

CAM Processor Logic:

The AM flip-flop controls when a CAM command is executed. AM is turned on at the end of F_0 of the CAM instruction fetch unless AM is already on from a previous CAM instruction. If AM is already on, the machine will go to F_3 and AM will be

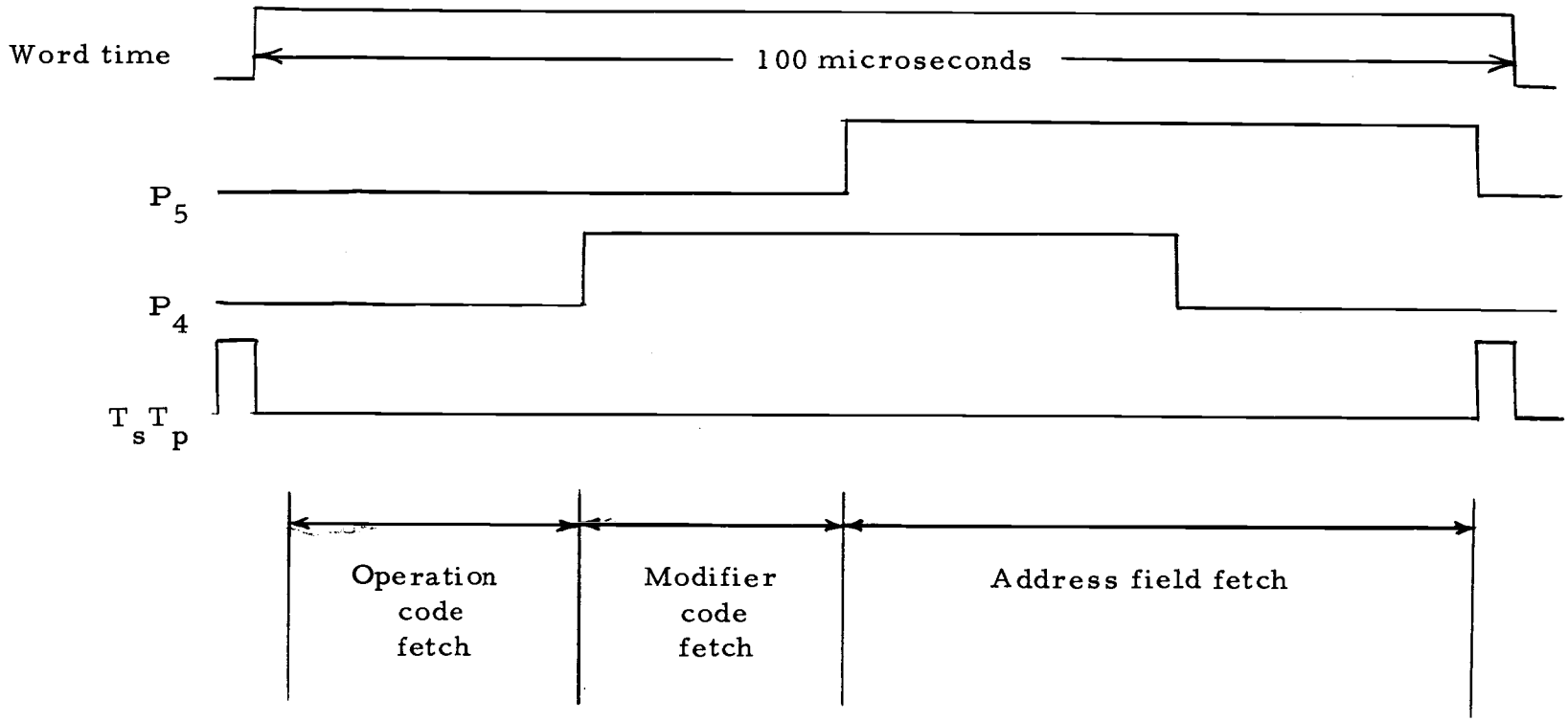


Figure 3.4. NEBULA processor timing.

turned on at the end of F_3.

AM Logic:

$$_1AM = T_s T_p F_0(CAM)\overline{AM} + T_s T_p F_3(CAM)$$

$$_0AM = T_s T_p(AM)$$

F-state Logic:

$$_1F_3 = T_s T_p(AM)F_0(CAM)$$

$$_0F_3 = T_s T_p(CAM)F_3$$

$$_1F_0 = T_s T_p(CAM)F_3$$

$$_0F_0 = T_s T_p(AM)F_0(CAM)$$

If $(\overline{AM})(CAM)F_0 = 1$ at $T_s T_p$, the F-state will remain as F_0.

The D register will copy the left half of C before fetching the next command.

D copy C logic:

$$D_{16} \leftarrow C_{17},\ D_i \leftarrow D_{i+1},\ i = 1, 2, \ldots, 15$$

during $F_0 P_5(\overline{AM})(CAM) + F_3 P_5(CAM)$.

C circulate logic:

$$C_{32} \leftarrow C_{17},\ C_i \leftarrow C_{i+1},\ i = 17, 18, \ldots, 31$$

during $F_0 P_5(\overline{AM})(CAM) + F_3 P_5(CAM)$.

The DI register copies memory during a CAM instruction fetch if AM is off and it copies D_1 during F_3. The K register is cleared while DI is being loaded to prepare for a parallel transfer from U to K.

DI load logic:

$$DI_{16} \leftarrow \text{Memory},\ DI_i \leftarrow DI_{i+1},\ i = 1, 2, \ldots, 15$$

during $F_0 P_5 (\overline{AM})(CAM)$

$$DI_{16} \leftarrow D_1,\ DI_i \leftarrow DI_{i+1},\ i = 1, 2, \ldots, 15$$

during $F_3 P_5 (CAM)$

Clear $K = F_0 P_5 (\overline{AM})(CAM) + F_3 P_5 (CAM)$

The instruction modifier field is always cleared during a CAM instruction fetch to prevent any address modification. Clear M logic:

$$M_8 \leftarrow 0,\ M_i \leftarrow M_{i+1},\ i = 1, 2, \ldots, 7$$

during $F_0 P_5 P_4 (CAM)$

During AM, the V register is held static.

If a CAM command is executed which calls for an LUA operation, then U circulates and clears in preparation for a parallel transfer from the high-speed search register (K).

$$U_{33} \leftarrow 0,\ U_i \leftarrow U_{i+1},\ i = 0, 1, \ldots, 32$$

during (AM)(LUA)

If a CAM command is executed which has ALL off, X will also circulate and load with zeroes. A parallel transfer from the CAM's address buffer will load the left half of X.

Clear X logic:

$$X_{33} \leftarrow 0,\ X_i \leftarrow X_{i+1},\ i = 0, 1, \ldots, 32$$

$$\text{during } (AM)(\overline{ALL})(\overline{OI_1})(\overline{OI_2})$$

The functions of OI_1 and OI_2 will be explained in the next section.

Random-Access Processor Logic:

The commands RAF, RASC, and RASS have many common processor logic functions.

During the second half of F_3, D shifts and is stored in DI; the left half of C circulates and is copied by D.

DI load logic:

$$DI_{16} \leftarrow D_1,\ DI_i \leftarrow DI_{i+1},\ i = 1, 2, \ldots, 15$$

$$\text{during } F_3P_5(RAF + RASS + RASC)$$

D shift and copy C logic:

$$D_{16} \leftarrow C_{17},\ D_i \leftarrow D_{i+1},\ i = 1, 2, \ldots, 15$$

$$\text{during } F_3P_5(RAF + RASS + RASC)$$

C circulate logic:

$$C_{32} \leftarrow C_{17},\ C_i \leftarrow C_{i+1},\ i = 17, 18, \ldots, 31$$

$$\text{during } F_3P_5(RAF + RASS + RASC)$$

The K register and the address buffer (BI) are cleared while DI is loading in preparation for a parallel transfer from U to K.

$$\text{Clear K, BI} = F_3P_5(RAF + RASS + RASC)$$

DI is transferred in parallel to the address buffer (BI) at the end of F_3.

Load BI (LBI) logic:

$$LBI = F_3T_sT_p(RAF + RASS + RASC)$$

AM is set at the end of F_3 and the random-access function is performed during the word time following F_3.

AM logic:

$${}_1AM = T_sT_pF_3(RAF + RASS + RASC)$$

$${}_0AM = T_sT_p(AM)$$

F logic:

$${}_0F_3 = T_sT_pF_3(RAF + RASS + RASC)$$

$${}_1F_0 = T_sT_pF_3(RAF + RASS + RASC)$$

OI_1 and OI_2 copy O_1 and O_2 during F_3 and are cleared during the middle of the next word time with AM off.

OI load logic:

$$OI_1 \leftarrow O_1,\ OI_2 \leftarrow O_2$$

$$\text{during } F_3P_5(\text{RAF} + \text{RASS} + \text{RASC})$$

Clear OI logic:

$$OI_1 \leftarrow 0,\ OI_2 \leftarrow 0$$

$$\text{during } (\overline{AM})P_4$$

When AM is one, the states of OI_1 and OI_2 define what type of command is being executed.

OI_1	OI_2	COMMAND
0	0	CAM
1	0	RAF
0	1	RASC
1	1	RASS

For a RAF instruction, U circulates and loads with zeroes during AM. This prepares U for a parallel transfer from the high-speed search register (K).

Clear U logic:

$$U_{33} \leftarrow 0,\ U_i \leftarrow U_{i+1},\ i = 0, 1, \ldots, 32$$

$$\text{during } (AM)(OI_1)(\overline{OI_2})$$

3.3.2 CAM Timing Logic

The execution of a CAM search requires one circulation time of the delay loops. This time is 100 microseconds and is the length of one word time of the NEBULA's processor. The word time of the execution is when the flip-flop AM is on. However, AM alone cannot be used to define the period of the CAM execution because AM may be on for a period ranging from 98 to 102 microseconds. This range is due to idiosyncracies in the NEBULA timing system.

An execution period which always exceeds 100 microseconds is required to insure that all of the CAM can be interrogated at least once during the CAM instruction execution. Additionally, a sequence of signals is required to

1) transfer U to the high-speed search register (K).

2) start the execution.

3) end the execution.

At least 100 microseconds is needed between 2) and 3).

The necessary timing can be achieved by the flip-flops, A_1, A_2, A_3, and the following logic.

$$_1A_1 = (AM)(\overline{Q_2})$$

$$_0A_1 = (\overline{AM})(Q_2)$$

$$_1A_2 = A_1$$

$$_0A_2 = \text{End}$$

$$_1A_3 = A_2$$

$$_0A_3 = \text{End}$$

Transfer U to K (TU) $= A_1\overline{A_2}$

Start $= A_1A_2\overline{A_3}$

End $= \overline{A_1}A_3$

The signal Q_2 is taken from the NEBULA system and when used as shown, will allow at least 101 microseconds for each CAM execution. The flip-flops are clocked J-K and with the logic shown, will produce the timing signals shown in Figure 3.5. A_1 through A_3 are clocked with the CAM 20 MHz clock. This produces a pulse width of about 50 nanoseconds for TU, Start, and End.

3.3.3 Delay Loop Logic

The logic of the delay loop system is relatively simple. The input of the delay line is driven by a write flip-flop (W_i) and the output of the delay line drives the read flip-flop (R_i). Information is maintained in storage by causing W_i to copy D_i which copies R_i. The flip-flop D_i is added to the loop to provide a one bit-time

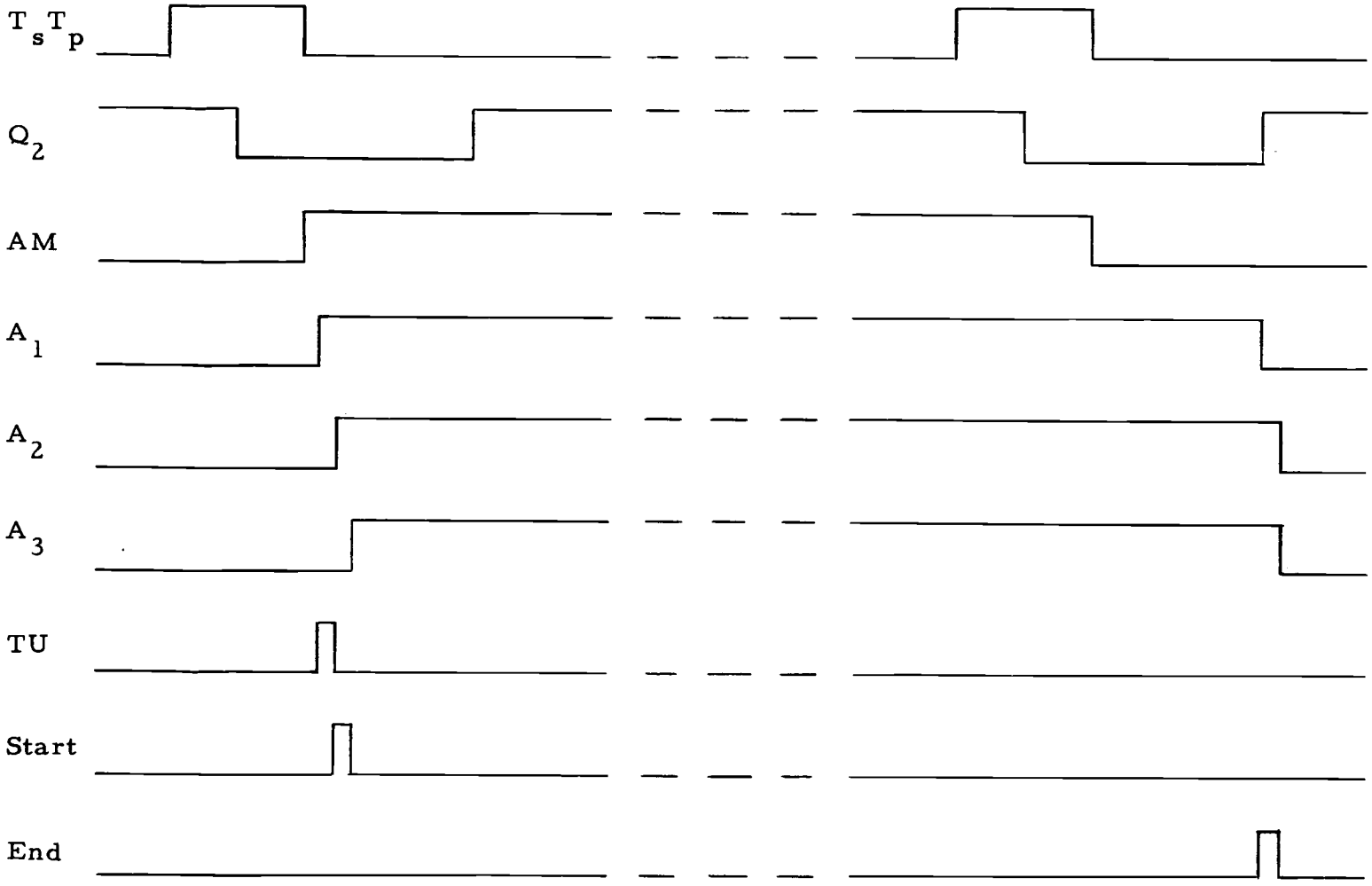


Figure 3.5. NEBULA CAM execution timing.

delay between observing information at R and copying it into W.

New information can be stored into the delay loop by causing W_i to copy the new information instead of D_i. Information will always be stored from the K register and it should be remembered that there are two types of storing; SUA causes all of K to be stored and SUS causes storage only at bit positions where the V register is one.

$$ {}_0W_i = \overline{S}\,\overline{D_i}\,(\overline{tag}) + \overline{D_i}[\overline{V_i}(SUS)(\overline{SUA})] $$

$$ + S\overline{K_i}[\overline{\overline{V_i}(SUS)(\overline{SUA})}] + Clear $$

$$ {}_1W_i = \overline{{}_0W_i} $$

The signal "S" is the store signal which is generated whenever an SUS or an SUA storage is to take place.

The signals "tag" and "clear" are used only for delay loop #33 which stores the tag bits.

Information can be read from the delay loop to the K register by simply causing K_i to copy D_i.

$$ {}_1K_i = D_i(KCR) $$

$$ {}_0K_i = \overline{D_i}(KCR) $$

The signal KCR is generated whenever K is to copy D.

The use-bit delay loop circulates information in a manner similar to the other delay loops but the write logic is much simpler.

$$_0W_u = \overline{S}_u\overline{D}_u + S_u\overline{K}_u$$

$$_1W_u = \overline{{}_0W_u}$$

The signal "S_u" stores K_u into the use-bit delay loop. K_u is a signal generated by the control logic.

3.3.4 Comparison Logic

Logic must be provided to compare the the K register with the delay loop output register (R) and to generate the equality signal (E) and the R greater than K signal (G).

$$\overline{E} = \sum_{i=0}^{33} V_i(R_i\overline{K}_i + \overline{R}_iK_i)$$

This function will generate the equality signal.

$$G = V_{33}R_{33}\overline{K}_{33} + \sum_{i=0}^{32} V_iR_i\overline{K}_i[\prod_{j=i+1}^{33} \overline{(V_j\overline{R}_jK_j)}]$$

This function will generate a greater-than signal which is accurate if all numbers are considered positive. However, NEBULA uses a two's complement number representation, with bit #32 being

the sign bit. In the two's complement system, negative numbers are "greater" than positive numbers. For example, + 1/2 is 0.100 and - 1/2 is 1.100.

Natural ordering is restored in the two's complement system by inverting the values of R_{32} and K_{32} in the greater-than function. This procedure also produces natural ordering for NEBULA floating-point numbers.

The greater-than function also presents a practical problem. It is a complex function which doesn't lend itself to straightforward implementation. The implementation of G was simplified by breaking the 34-bit word into six groups of four bits and two groups of five bits. A greater-than signal (G_i) and an equal signal (E_i) are generated for each of these eight groups. Consideration of these signals yields G.

$$G = G_i + \sum_{i=1}^{7} G_i \left[\prod_{j=i+1}^{8} E_j \right]$$

This approach allows implementation using practical levels of fan-in and fan-out for the gates. This function demands a maximum fan-in and fan-out of eight. However, the signals from R and K must now propagate through five gates before they can affect the value of G.

3.3.5 Standard Addressing

The address counter (AC) in Figure 3.6 is an 11-bit binary counter which is synchronized with the delay loops so that it has a unique state for each word in the delay loop system.

For the random-access instructions, the address buffer (BI) is loaded from DI with the address and a comparison between BI and AC will generate an address-equal signal (AE) which is used to strobe the addressed word as it appears in the delay loop register (D).

For CAM instructions, the addressing system is used for two purposes.

1. When ALL = 0, operation will take place on only one word in the CAM. When the search finds this word, the signal ARO will store the contents of AC in the address buffer (BI) until the end of the word time. The signal MRO will gate the address buffer into the left half of the X register.
2. When ALL = 1, operation may take place on all words which satisfy the search criteria. However, if the CAM instructions were left in operation for the 101 microseconds between Start and End, some words may be operated on twice. The addressing system is used to prevent this

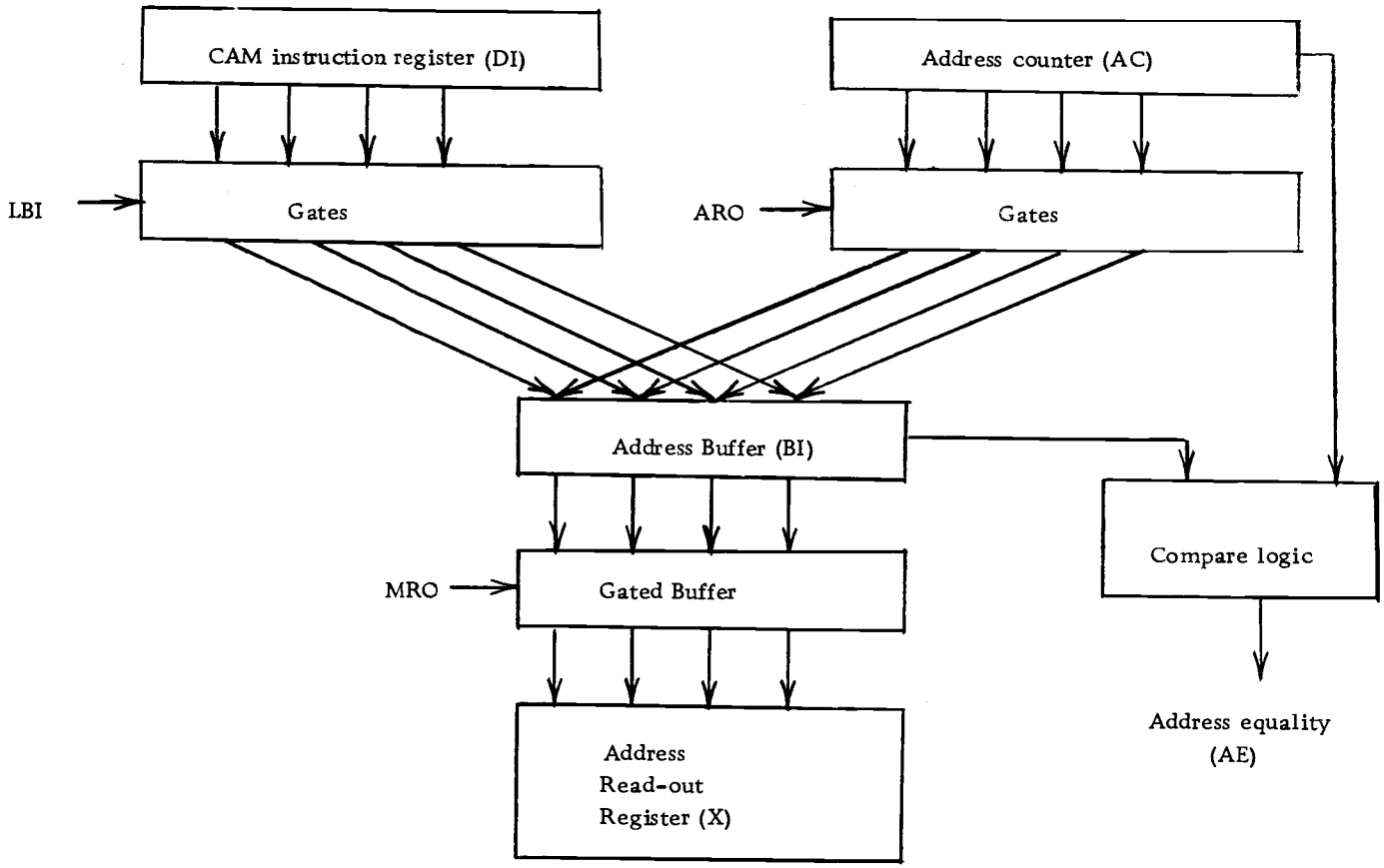


Figure 3.6. NEBULA CAM random-access addressing system.

situation. When the execution begins, the contents of AC is transferred to BI to store the address of the first word observed in the execution. The address counter will not equal BI until all words have been observed once by the comparison logic. When AE equals one, the instruction execution is terminated.

Logic for the standard addressing system will be explained with the control logic.

3.3.6 Control Logic

The control logic must monitor the instruction being executed and control the operations of the system. A flip-flop P is used to control the execution of the instructions.

All instructions are initiated from the NEBULA processor by

1. loading DI and OI with the instruction information,
2. clearing K and BI in preparation for a parallel transfer between U and K and, for random-access commands, a parallel transfer from DI to BI, and
3. turning AM on to begin the CAM timing sequence described in Section 3.3.2.

The CAM timing sequence generates TU to transfer U to K and then generates Start which turns P on to begin execution of the instruction.

A second flip-flop (C) is turned on at the Start and if C is on when the signal End occurs, then C_9 is set in the NEBULA processor.

$$_1P = \text{Start}$$

$$_1C = \text{Start}$$

$$\text{Set } C_9 = (C)\ \text{End}$$

For random-access instructions, P is left on until the End and while P is on, signals are generated to operate appropriately at the address that equals the address buffer. The function S controls storing and KCR controls loading K.

$$_0P = \text{End}$$

$$S = (AE)(OI_2)P$$

$$S_u = (AE)(OI_2)P$$

$$K_u = (OI_1)(OI_2)$$

$$KCR = (AE)(OI_1)(\overline{OI_2})P$$

C is turned off immediately for RASC and RASS, and for RAF, C is turned off if the use bit is on at the word addressed.

$$_0C = OI_2 + (AE)(OI_1)(\overline{OI_2})(D_u)$$

K is transferred back to U for the RAF instruction.

$$TK = (OI_1)(\overline{OI_2})\ \text{End}$$

Notice that the signal AE is not generated until one bit time after the address counter (AC) equals the address buffer (BI). This delay is due to the logic gate delays in the generation of AE.

For CAM instructions, each word at R is compared with the search register (K) and if the word satisfies the search criteria, the operate function (Op) will be one when the word passes between D and W. The word cannot be operated on between R and D because one bit-time is not of sufficient length to allow signals to propagate through the comparison logic, control logic and through the operation drivers.

$$Op = P\ (\overline{OI_1})(\overline{OI_2})[(\overline{TAG}) + R_{33}]\cdot$$
$$[(USE)(R_u) + (\overline{USE})(R_u)]\cdot$$
$$[E(EQL) + G\,(GTR) + (\overline{E})(\overline{G})(LES) + (\overline{GTR})(\overline{EQL})(\overline{LES})]$$

The signal paths generating this function are adjusted so that operation will occur when the word is being transferred from D to W.

$$S = [(SUA) + (SUS)]\ (Op)$$

$$S_u = (CHU)(Op)$$

$$K_u = (\overline{USE})(CHU)$$

$$KCR = (LUA)(Op)$$

C is turned off as soon as a word is found which satisfies the search criteria.

$${}_0C = (Op)$$

When ALL is off, P will be turned off as soon as one word is operated on.

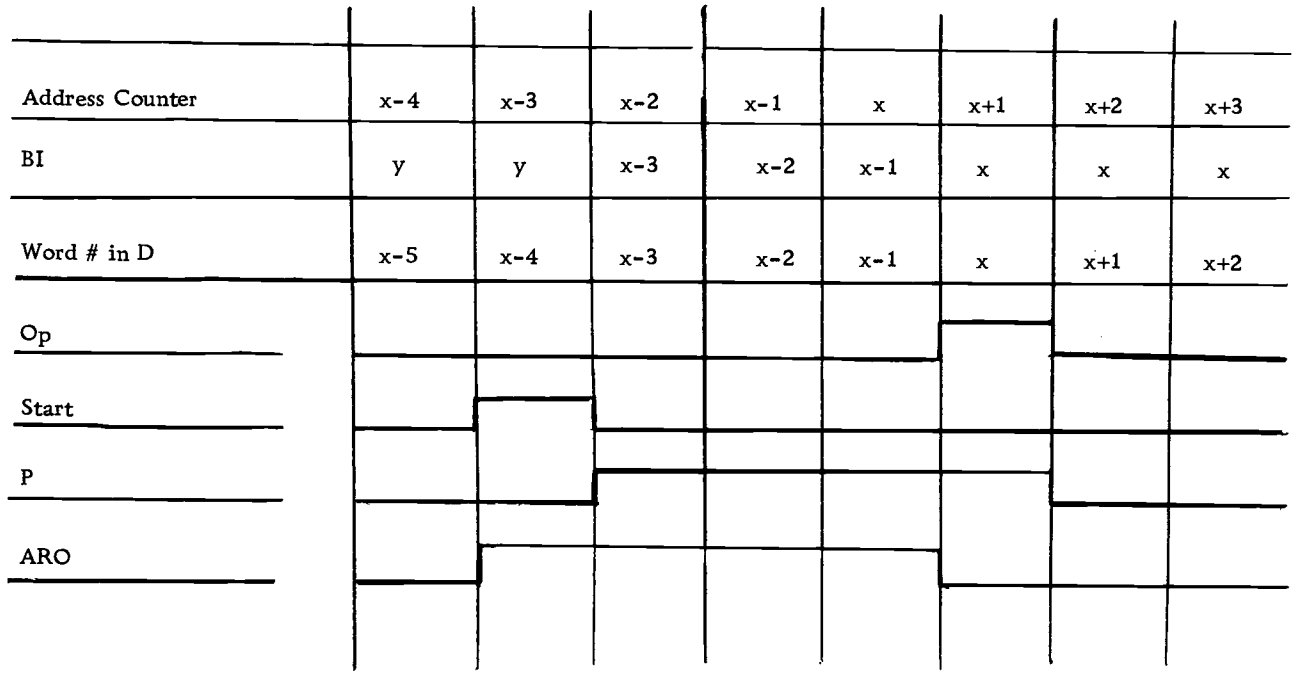
$${}_0P = (\overline{ALL})(Op)$$

Also, the ARO signal is kept one until Op becomes one. BI will then hold the address of the word which was operated on. If C is off, indicating that a word was in fact found which satisfied the search criteria, BI is transferred to X at the End.

$$ARO = (\overline{ALL})(\overline{OI_1})(\overline{OI_2})(Start)$$
$$+ (\overline{ALL})(\overline{OI_1})(\overline{OI_2})(\overline{Op})P$$

$$MRO = (\overline{ALL})(\overline{OI_1})(\overline{OI_2})\overline{C}(End)$$

Figure 3.7 illustrates the timing for this logic.



Operation is on word #x and BI is loaded with the address of x.

Figure 3.7. $\overline{\text{ALL}}$ timing for address read-out (ARO).

When ALL is on, it is desired to leave P on for exactly one circulation time. As explained in section 3.3.5, BI is loaded with the starting address and P is turned off before that address is operated on again. Figure 3.8 illustrates the timing for this logic.

$$ARO = (ALL)(TU)$$

$$_0P = (ALL)(AE)$$

Tagging and tag clearing can be performed when a word satisfies the search criteria.

$$tag = (SST)(Op)$$

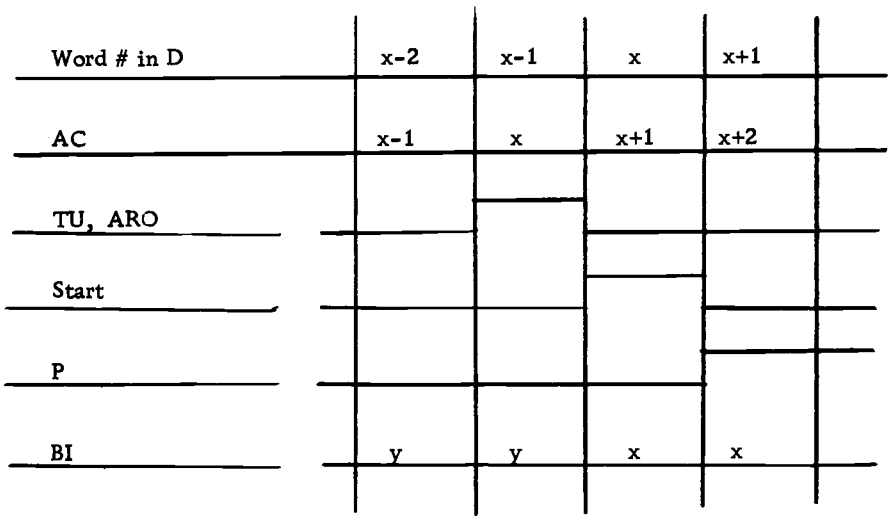
$$clear = (CST)(Op)$$

If the operate function (Op) is delayed one bit time to generate (Op+) the next word can be tagged.

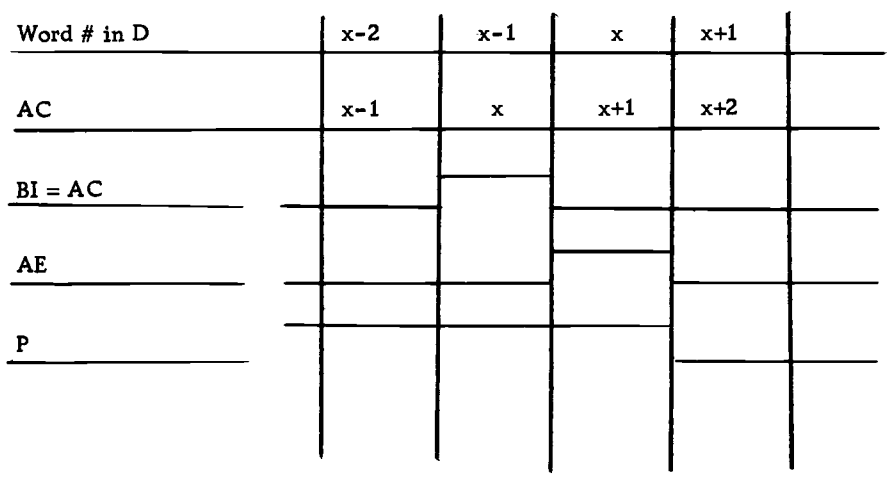
$$tag = SNT\ (Op+)$$

$$clear = CNT\ (Op+)$$

Similarly, if when a word is in R that satisfies the search criteria, the previous word, which is in D can be tagged if the decision to tag can be made in one bit time. Limitations of the logic speed would only allow this operation for the search criteria of equality and use-bit and tag-bit conditions.



Operation begins at word #x+1; BI holds address x.



P turns off after AE to prevent a second operation on word #x+1.

Figure 3.8. ALL timing for control flip-flop P.

$$\text{tag} = (SPT)({}_1P + P)({}_0\overline{P})(E + \overline{EQL})[\overline{TAG} + R_{33}]$$

$$[(USE)(R_u) + (\overline{USE})(R_u)]$$

$$\text{clear} = (CPT)({}_1P + P)({}_0\overline{P})(E + \overline{EQL})(\overline{TAG} + R_{33})$$

$$[(USE)(R_u) + (\overline{USE})(R_u)]$$

This special logic performs the tagging operation, but P is still turned off by the Op function.

Special logic must be used when (LUA)(ALL) is specified. This calls for U to be loaded with all locations. This is not reasonable so when (LUA)(ALL) is on, Op is inhibited.

However, (LUA)(ALL)(GTR) is used to fetch the largest word. Special logic must be used to perform this task because of the one bit-time delay between comparisons and operation.

Consider the case when K holds the number 20 and R holds 40. The comparison will show R greater than K and when the 40 gets to D, it will be loaded into K. Now in the next bit time, the 40 will move to D and suppose R loads with the number 30. K still holds 20 and the comparison logic will again show R greater than K and will inform the control logic that the 30 should be loaded into K when it reaches D. Thus in the next bit time, K will load with 40 and then the following bit time will load 30 from D into K. This procedure is inaccurate since K

should hold the largest word but it in fact holds 30 and not 40.

This difficulty can be overcome by comparing R with K and also comparing R with the word before it which is in D. For (LUA)(ALL)(GTR), K will be loaded only if the word in R is found to be greater than K and also greater than D.

$$KCR = (LUA)(ALL)(GTR)(\overline{LES})[\overline{TAG} + R_{33}][(USE)R_u + (\overline{USE})(\overline{R_u})]$$

$$(G)[\overline{TAG} + D_{33}][(USE)(D_u) + (\overline{USE})(\overline{D_u})](G_D)P$$

G is the function for R greater than K and G_D is the function for R greater than D.

Similar logic is used for (LUA)(ALL)(LES).

3.4 Hardware Implementation

3.4.1 Electronics

The NEBULA CAM system was constructed from monolithic integrated circuits. There are several families of logic circuits to choose from the integrated circuit technology, but the principle factor in the choice for this system was speed. In order to implement the comparison and control logic for operation at the 20 MHz clock rate, logic gates with propagation delays of under 10 nanoseconds were required.

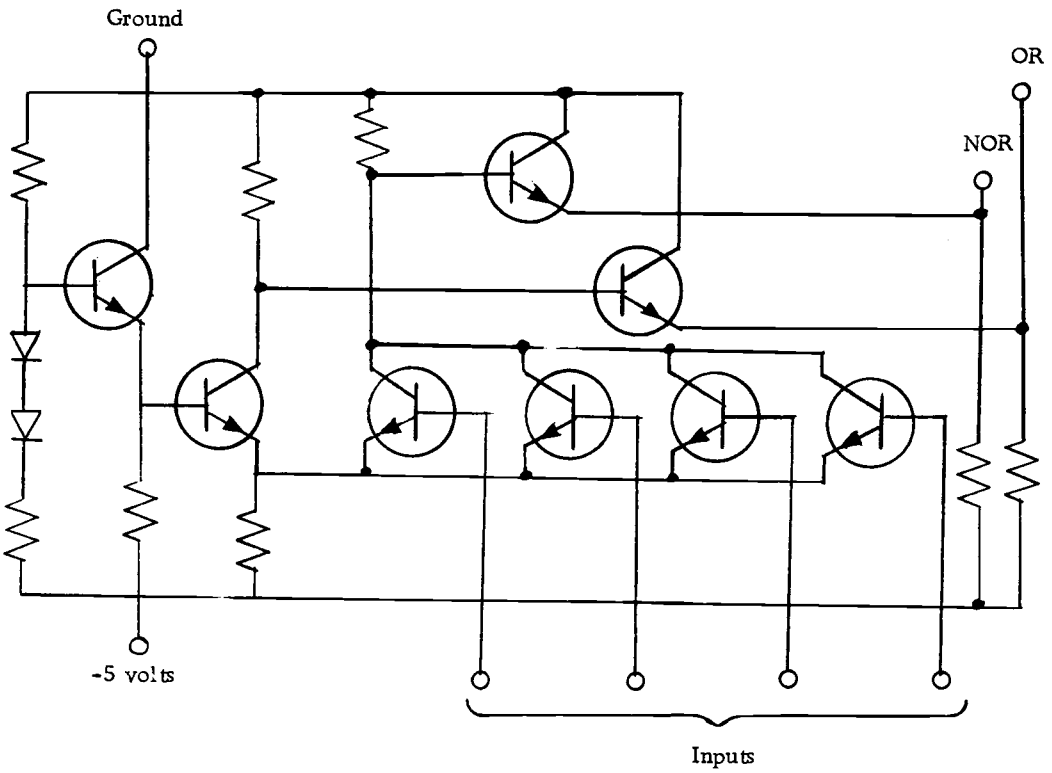
The NEBULA CAM system uses Motorola Semiconductor's

version of emitter-coupled logic (MECL). This is a non-saturating logic form which can provide NOR or OR logic functions in typically five nanoseconds. The units which were purchased exhibited propagation delays very close (within one nanosecond) to the typical figure. These circuits were the fastest that were commercially available. The MECL logic also has a speed advantage over other logic forms because there is no requirement for inverters. The NOR-OR outputs of the MECL gates provide true or false functions with equal speed.

The circuit schematic for a 4-input MECL gate is shown in Figure 3.9. Both the NOR and OR outputs are driven by emitter followers which offer a low output impedance. Also, the input impedance of the circuit is very high since the input is the base of a transistor. This combination of low output impedance and high input impedance allows any gate to drive many other gates without the need of special driver circuits. All MECL gate outputs have a specified minimum fan-out of 15.

The MECL circuits operate from a single power supply voltage of -5 volts and the signal levels are typically -0.8 volts for a logical one and -1.6 volts for a logical zero. The voltage difference between a one and a zero state is only 0.8 volts which eases the problem of charging stray circuit capacitances.

With any high-speed voltage transition, the problem of overshoot and ringing is experienced if straight wire connections are used



Typical "1" voltage -0.8 volts
Typical "0" voltage -1.6 volts
Typical propagation delay 5 nanoseconds

Figure 3.9. The circuitry for a MECL 4-input NOR-OR gate.

for distances of more than six inches. This problem can be overcome by the use of twisted-pair transmission lines. However, the twisted-pair must be terminated with approximately 130 ohms in order to prevent reflections within the line. The MECL logic circuits will drive twisted-pair lines when terminated with either:

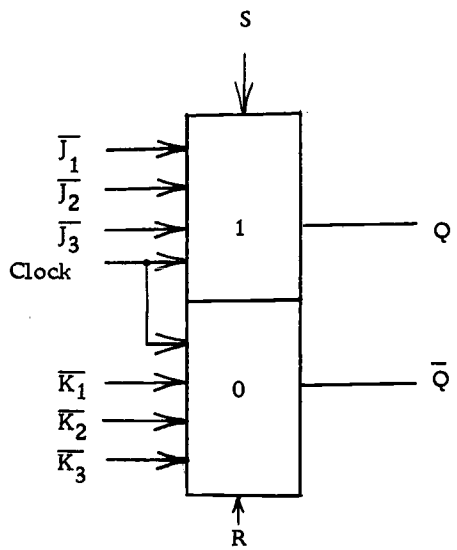
1) 130 ohms to -1.5 volts.

2) 130 ohms in series with 100 pf. to ground.

In some cases, the MECL gates proved to be too fast for the timing requirements of the system. At some points in the logic, the propagation delay had to be lengthened by capacitively loading the gate outputs. This technique was used to add up to 20 nanoseconds to the gate propagation delay.

The MECL flip-flops are J-K type flip-flops which are triggered by a positive-going signal. The clock pulse which is used in the NEBULA CAM is one for 35 nanoseconds and zero for 15 nanoseconds. The flip-flop is triggered on the zero-to-one transition and the inputs must be stable while the clock is in the zero state. Figure 3.10 gives a graphical representation of the MECL flip-flop and illustrates the truth-tables for the clocked inputs and the direct inputs.

The CAM system electronics is principally constructed from MECL logic gates, but some special circuitry is required in the delay loop electronics. Figure 3.11 shows the circuitry used in the delay loop.



Clocked Inputs

$\overline{J_1} + \overline{J_2} + \overline{J_3}$	$\overline{K_1} + \overline{K_2} + \overline{K_3}$	Q_n	Q_{n+1}
0	0	Q	$\overline{Q}$
1	0	Q	0
0	1	Q	1
1	1	Q	no change

Direct Inputs

R	S	Q
0	0	no change
0	1	1
1	0	0
1	1	?

Figure 3.10. The MECL flip-flop.

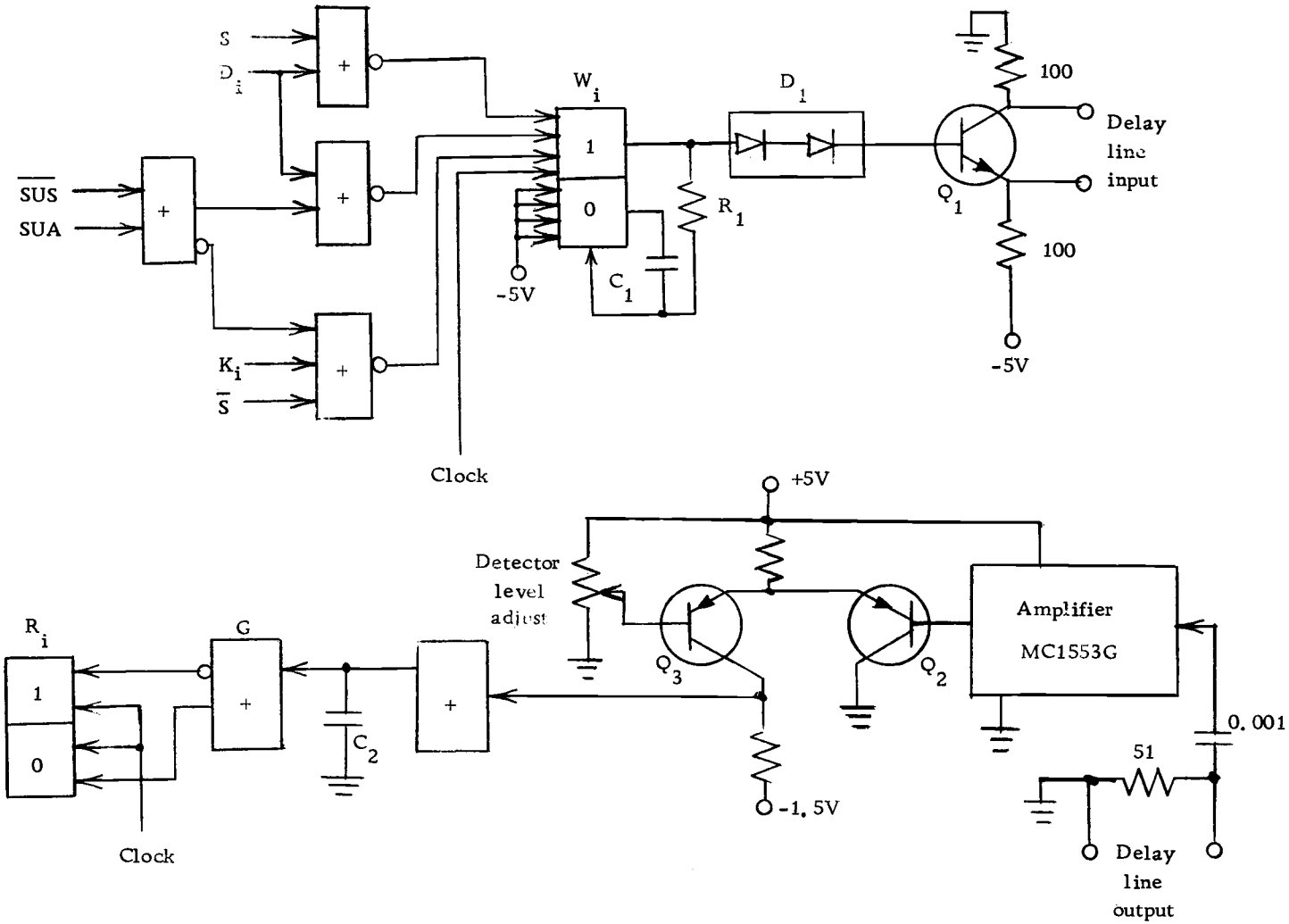


Figure 3.11. NEBULA CAM delay loop electronics.

The logic at the input of W_i will set W_i when a one bit is to be stored in the delay line. However, the combination R_1 and C_1 will allow W_i to remain set for only 25 nanoseconds before returning to the zero state. Therefore, the output of W_i is used as the source of return-to-zero (RZ) signals for the delay line. A 25 nanosecond pulse stores a one, and no pulse is a zero.

The output of W_i is voltage shifted -1.6 volts by the diode D_1 before driving Q_1. The emitter and collector of Q_1 combine to provide a balanced drive to the delay line input. The emitter of Q_1 rises 0.8 volts and the collector drops 0.8 volts. The glass delay lines used with the NEBULA CAM system have a signal attenuation ranging from 39 db to 45 db. This signal loss is restored at the delay line output by the amplifier MC1553G. The MC1553G is an integrated-circuit video amplifier built by Motorola Semiconductor. The MC1553G is feedback stabilized so that the d.c. output voltage is stable within ± 0.05 volts and the voltage gain is 52 ± 0.2 db. The bandwidth of the MC1553G is typically 35MHz.

The transistors Q_2 and Q_3 are used to form a level detector for the output signal. The output of Q_3 drives a MECL gate and the capacitor C_2 widens the 20 nanosecond output pulse to 50 nanoseconds before being applied to the inputs of R_i.

The output of G must change states only when the clock is in the one state. This can be insured if the clock frequency is

crystal controlled to within ± 0.001% and the delay lines delay are guaranteed 100.000 ± 0.013 microseconds over a temperature range $0^{o}C$ to $50^{o}C$.

The 0.001% frequency variation amounts to ± 1 nanosecond in the 100 microseconds required for a pulse to travel through the delay line. This two nanosecond clock variation plus the 26 nanosecond variation in pulse delay gives a possible total variation of 28 nanoseconds between the delay line output pulse and the clock pulse. This 28 nanosecond variation is absorbed within the 35 nanoseconds in which the clock is in the one state.

3.4.2 Packaging

The NEBULA CAM system was constructed on circuit boards with edge-card connections and mounted in a standard 19 inch relay rack in two $5\frac{1}{4}$ inch high card cages. The 35 delay lines with their loop electronics were each mounted on a $4\frac{1}{2}$ inch by 9 inch two-sided, printed-circuit board. The other circuitry was mounted and hand wired on 22 $4\frac{1}{2}$ inch by $4\frac{1}{2}$ inch "universal" printed circuit boards.

The power requirements of the system were as follows:

-5 volts - 18 Ampere

-1.5 volts - 1 Ampere

+5 volts - 1 Ampere

3.4.3 Testing

The testing of the CAM system is facilitated by the use of several diagnostic programs which allow the NEBULA processor to exercise the CAM functions and to aid in the location of faculty circuitry, timing, or wiring. The diagnostic programs first establish the proper functioning of the random-access functions and then proceed to check out the content-addressing functions. If an error is detected, the operator is referred to a table showing possible failure points in the system.

4. SYSTEM EVALUATION

4.1 System Cost

The cost of the NEBULA CAM system hardware can be divided into the following categories:

1) 35 glass delay lines	$5,250
2) Loop electronics	$2,300
3) Random-access addressing	$ 500
4) Comparison logic	$ 900
5) Control logic	$ 350
6) Packaging	$ 100
7) Power supplies	$ 500

This gives a total hardware cost of nearly $10,000 for a system which stores 2048 x 35 = 71,680 bits of information. This represents a cost/bit of about 14 cents.

If this same system were implemented as a random-access system only, the total hardware cost would be reduced to about $8,700 by elimination of the comparison logic and portions of the control logic. The cost for the random-access system would then be 12 cents per bit.

This comparison illustrates that the addition of the content-addressing capability to this circulating memory requires only a 15% increase in the hardware cost. This is in contrast to a static type

memory system which would require comparison logic at every memory location. Very few CAM systems are known to have actually been constructed, but the Goodyear Aerospace Corporation developed and is constructing a 2048 x 48-bit magnetic core CAM system under the Air Force contract AF 30(602)3549. This contract was for $614,000 and Goodyear added an equivalent amount to give a total development and construction cost of $1,200,000.[1] This system searches for equality in 10 μsec and for maximum in 40 μsec and can define up to eight variable-size search fields (10). A 2048 x 48 bit random-access magnetic core memory system can be purchased from Fabri-Tek Inc., for under $11,000.[2] These figures serve to illustrate that the static content-addressed memory is far more expensive than the static random-access system.

The Goodyear CAM system is faster and more versatile than the NEBULA CAM system but the Goodyear system is much more expensive. The NEBULA CAM system was developed under a $25,000 grant from the Office of Naval Research.

---

[1] Previte, Jim. Engineer, Rome Air Development Center, Personal Communication, Rome, New York. January 23, 1968.

[2] Okerlund, Richard. Sales Representative, Fabri-Tek Inc., Personal Communication, Los Angeles, California. January 24, 1968.

4.2 Command Set Versatility

The NEBULA CAM system's command set allows a wide variety of searches to be performed. In addition to use-bit and tag-bit conditions, masked searches can be made directly for:

1) Equal

2) Greater

3) Less

4) Greater or Equal

5) Less or Equal

6) Not Equal (Greater or Less)

7) Largest

8) Least

By sequencing instructions, it is also possible to easily perform searches for:

1) Next greater

2) Next smaller

3) Between limits

There is also a variety of operations that can be performed. Storing and fetching can be performed simultaneously with any desired combination of tagging and use-bit operations.

Ordered retrieval of a list can be easily accomplished using the following procedure:

1) Tag the words in the list.

2) Fetch the smallest tagged word.

3) Clear the tag on the word fetched.

4) Fetch the smallest tagged word.

5) etc., until all the list has been fetched.

The Store U Selectively (SUS) command provides a great deal of flexibility in the operations that can be performed. This command has been shown (15) to allow any logical function to be performed on the memory words.

For example, it is possible to perform addition of a constant to all memory locations in a serial-by-bit and parallel-by-word procedure. First, all words are tagged which should have zeroes written at bit #1, the zeroes are written with a SUS command and then words are tagged which need ones written at bit #1. The ones are written and then the sequence is repeated for bits #2 through #32.

Using this procedure, a constant can be added to 2048 words with a routine requiring about 20 instructions. The total addition can be performed with about 320 instruction executions for a 32-bit word. This is a speed advantage over a random-access system which would require only a five instruction routine but 10,000 instruction executions to perform the same task. So for this specific application, the CAM system offers a speed advantage but at the sacrifice of programming simplicity.

4.3 Applications

The capability of content-addressing is applicable to a variety of problems but only a few will be discussed here.

An immediate application for a CAM is in the construction and searching of symbol tables in assemblers and compilers. In the NEBULA assembler, STAR, symbol tables are constructed using a folded-sum function to compute a "hash" address. This technique requires a sub-routine using 30 instructions to store a symbol and a 32 instruction routine to search the table and fetch a value. With the CAM instructions, it is possible to store 30-bit symbols and their 32-bit values at consecutive locations in the CAM with a six instruction routine and fetch a value with an eight instruction routine. This illustrates a significant improvement in programming simplicity and also in speed.

The standard NEBULA symbol store routine requires 50 word times plus 8 word times for each time a word is addressed to a location which is already in use. The CAM routine requires only 12 word times. Also, the CAM symbol table can be easily output in a pseudo-alphabetic order by using an ordered retrieval of the symbols.

A CAM system can also be useful in pattern recognition problems. By way of illustration, Yau and Yang (20) suggest a method for high-speed handwritten character recognition. This method calls for

comparison of 64 bits of information against 35 stored templates. The NEBULA CAM system can directly implement solution to this problem by storing two halves of each template in successive CAM memory locations with the bit #0 being set for the first half of each template. The input pattern can then be classified by first comparing the first 32 bits of the pattern with the locations that have bit #0 in the one state. The SNT command can set the tag on the second half of all templates which "fit" on the first half. A second CAM command can compare the second half of the input pattern with tagged template halves to find which templates matched over both halves.

The CAM solution to this problem requires only a few instructions but a random-access system would require a linear search of the templates with masked comparisons after each fetch. The value of the CAM system would be even greater if the number of stored templates was larger.

4.4 Limitations

For certain classes of problems, the size of the NEBULA CAM system would prove limiting. For example, the search capabilities of a CAM system would be an advantage in the implementation of written text editing routines. However, for problems such as spelling corrections, a dictionary of about 10,000 words would be necessary where the word length may require up to 96 bits per word. The

2048-word NEBULA CAM is too small in word length and number of words for efficient implementation of such a problem.

However, even for smaller scale problems, the word length in the NEBULA CAM is limiting when several sets of data are stored in the CAM. A larger tag field on the NEBULA system would facilitate separation of the various sets of data.

Another limitation appears when for example it is desired to locate the first even numbered location matching a given set of search criteria. Problems of this nature could be solved easily by incorporating a masked equality search over the field of the 11-bit address as one of the search criteria.

Also, if ordering of information is desired, the fact that the NEBULA CAM system begins its searches at random locations proves to be a burden. It is often more convenient for searches to always begin at the same known location.

Finally, the NEBULA CAM system is limited in speed. With its ability to search and operate on 2048 words in 100 microseconds, it is very fast for use with the NEBULA system. However, the use of this 100 microsecond system with a high-speed parallel processor would not be extremely useful except for a few special applications. A high-speed parallel processor can execute up to 100 instructions in this 100 microsecond period.

The speed of a delay line CAM system can be improved by

shortening the circulation time. In order to maintain the same memory capacity, however, this requires an equivalent increase in the rate of information flow. As higher speed delay lines and logic gates become available, the CAM system speed can be increased with no sacrifice in memory capacity. With a 100 MHz information rate, 2048 words can be searched in 20 microseconds.

The size limitations of the NEBULA CAM system could be overcome easily. The word length of the system could be very easily increased by simply adding more storage delay lines. The control system would be unchanged in principle. The word capacity could be increased by several methods:

1. Increasing the information rate while keeping the same circulation time.
2. Increasing the circulation time while keeping the information rate constant.
3. Paralleling two or more delay line systems.

This discussion illustrates that most of the CAM system's limitations can be eliminated. The speed of the circulating system can be improved upon, but because the circulating system is serial in nature, static memory systems will probably always be faster.

4.5 Conclusions

The NEBULA CAM system has been shown to be a relatively

low cost system with a reasonable degree of versatility. For certain classes of problems, the CAM system offers an improvement over the random-access solutions to these problems. This improvement is represented as a processing speed improvement and in some cases, as an improvement in programming simplicity.

The NEBULA CAM system has some limitations in capacity and tag-field length but the system is powerful enough to be useful as a research tool for further investigation into other content-addressing applications.

BIBLIOGRAPHY

1. Boles, J.A. The logical design of the NEBULA computer. Ph.D. Thesis. Corvallis, Oregon State University, 1968. 150 numb. leaves.

2. Boles, J A., P.T. Rux and F.W. Weingarten. NEBULA: A digital computer using a 20 Mc glass delay line memory. Communications of the Association for Computing Machinery 9:500-508. 1966.

3. Crofut, W.Q. and M.R. Sottile. Design techniques of a delay-line content-addressed memory. IEEE Transactions on Electronic Computers EC-15:529-534. 1966.

4. Davies, Paul. A superconductive associative memory. In: Proceedings of the Spring Joint Computer Conference of the American Federation of Information Processing Societies, San Francisco, 1962. Palo Alto, National Press, 1962. p. 79-88.

5. Estrin, G. and R.H. Fuller. Some applications for content-addressed memories. In: Proceedings of the Fall Joint Computer Conference of the American Federation of Information Processing Societies, Las Vegas, 1963. Baltimore, Spartan Books, 1963. p. 495-508.

6. Falkoff, A.D. Algorithms for parallel-search memories. Journal of the Association for Computing Machinery 9:488-510. 1962.

7. Frei, E.H. and J. Goldberg. A method of resolving multiple responses in a parallel search file. IRE Transactions on Electronic Computers EC-10:718-722. 1961.

8. Hanlon, A.G. Content-addressable and associative memory systems, a survey. IEEE Transactions on Electronic Computers EC-15:509-521. 1966.

9. Kaplan, Albert. A search memory subsystem for a general-purpose computer. In: Proceedings of the Fall Joint Computer Conference of the American Federation of Information Processing Societies, Las Vegas, 1963. Baltimore, Spartan Books, 1963. p. 193-200.

10. Knapp, Morris A. RADC programs in associative processing. A paper presented at a Seminar on Associative Processing, Washington, D.C., 1967. Sponsered by the Office of Naval Research and the Rome Air Development Center. n.d. 16 p.

11. Kiseda, J.R. et al. A magnetic associative memory. International Business Machines Journal of Research and Development 5:106-121. 1961.

12. Lee, C.Y. and M.C. Paull. A content addressable distributed logic memory with applications to information retrieval. Proceedings of the Institute of Electrical and Electronic Engineers 5:924-932. 1963.

13. Lewin, Morton H. Retrieval of ordered lists from a content-addressed memory. Radio Corporation of America Review 23: 215-229. 1962.

14. Nickodemus, W.A. (ed.). Progress report on the NEBULA computer, January 14, 1966. 79 numb. leaves. (Oregon State University, Computer Center. Office of Naval Research Contract NONR-1286(11))

15. Porter, Sigmund N. Use of multiwrite for general programmability of search memories. Journal of the Association for Computing Machinery 13:369-373. 1966.

16. Rux, P.T., F.W. Weingarton and F.H. Young. Algorithms for circulating associative memories. Corvallis, 1966. 15 p. (Oregon State University. Computer Center. Report no. 66-4)

17. Seeber, R.R. and A.B. Lindquist. Associative memory with ordered retrieval. International Business Machines Journal of Research and Development 6:126-136. 1962.

18. Weingarten, F.W., P.T. Rux and J.A. Boles. On an associative memory for the NEBULA computer. Corvallis, Oregon State University, Department of Mathematics, 1964. (Mimeographed report)

19. Yau, S.S. and C.C Yang. Pattern recognition by using an associative memory. IEEE Transactions on Electronic Computers EC-15:944-947. 1966.

20. Young, F.H. Circulating associative memories. Corvallis, Oregon State University, Department of Mathematics, 1962. (Mimeographed report)