STAN-CS-71-220

SU-SEL-71-009

Dynamic Memories with Enhanced Data Access

by

Harold S. Stone

February 1971

Technical Report No. 14

This work was supported wholly by the Joint Services Electronics Program (U.S. Army, U.S. Navy, and U.S. Air Force) under Contract N00014-67-A-0112-0044

DIGITAL SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES

STANFORD UNIVERSITY · STANFORD, CALIFORNIA

34

SION for

CFSTI	WHITE SECTION [✓]
DDC	BUFF SECTION ☐
UNANNOUNCED	☐

JUSTIFICATION..............................

....

BY...

DISTRIBUTION/AVAILABILITY CODES

DIST.	AVAIL. and/or SPECIAL	
A		

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Digital Systems Laboratory	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP

3. REPORT TITLE

DYNAMIC MEMORIES WITH ENHANCED DATA ACCESS

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Technical Report no. 14 February 1971

5. AUTHOR(S) (Last name, first name, initial)

Stone, Harold S.

6. REPORT DATE February 1971	7a. TOTAL NO. OF PAGES 31	7b. NO. OF REFS 6
8a. CONTRACT OR GRANT NO. N-00014-67-A-0112-0044	9a. ORIGINATOR'S REPORT NUMBER(S) SEL 71-009; STAN-CS-71-220	
b. PROJECT NO. 6960		
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
d.		

10. AVAILABILITY/LIMITATION NOTICES

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Joint Services Electronics Program: U.S. Army, U.S. Navy, and U.S. Air Force

13. ABSTRACT

Dynamic memories are commonly constructed as circulating shift registers, and thus have access times that are proportional to the size of memory. When each word in a dynamic memory is connected to r words, $r \geq 2$, access time can be proportional to the base r logarithm of the size of memory. This paper describes a memory that achieves minimum access time for $r = 2$. The memory can also be operated in an efficient binary search mode. Slight variations of the interconnection patterns lead to a memory that is well suited for FFT and certain matrix computations.

DD FORM 1 JAN 64 1473

Unclassified

Security Classification

Unclassified

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
dynamic memories						
MOS shift registers						
magnetic bubble memories						
perfect shuffle						
binary search						
access time						

Security Classification

STAN-CS-71-220

SEL-71-009

DYNAMIC MEMORIES WITH ENHANCED DATA ACCESS

by

Harold S. Stone

February 1971

Technical Report No. 14

DIGITAL SYSTEMS LABORATORY

Stanford Electronics Laboratories

Stanford University

Stanford, California

This work was supported wholly by the
Joint Services Electronics Program
(U.S. Army, U.S. Navy, and U.S. Air Force)
under Contract N00014-67-A-0112-0044

Digital Systems Laboratory
Stanford Electronics Laboratories

Technical Report No. 14

January 1971

Dynamic Memories with Enhanced Data Access

by

Harold S. Stone

Abstract

Dynamic memories are commonly constructed as circulating shift registers, and thus have access times that are proportional to the size of memory. When each word in a dynamic memory is connected to r words, $r \geq 2$, access time can be proportional to the base r logarithm of the size of memory. This paper describes a memory that achieves minimum access time for $r = 2$. The memory can also be operated in an efficient binary search mode. Slight variations of the interconnection patterns lead to a memory that is well suited for FFT and certain matrix computations.

I. Introduction

In some memory technologies, the storage medium inherently requires that there be a steady-state circulation of data. Examples of such memories include magnetic drums and disks, MOS shift registers, and magnetic bubble memories. In this paper, we shall refer to such memories as dynamic memories.

For practical reasons, data movement in dynamic memories is normally cyclic. In the case of the magnetic drum, data is stored on the circumference of the drum, so that the rotation of the drum relative to a fixed head produces the cyclical movement of the data. MOS shift-register memories are commonly constructed as circulating shift registers although there is no constraint that forces such memories to use the cyclic interconnection pattern.

Given the constraint that data must be moved continuously in a dynamic memory, the cyclical structure of the memory causes difficulty in achieving simultaneously both a large storage capacity and a short access time.

In a cyclic memory, the access time to a randomly selected item increases linearly with the size of the memory. In this paper we investigate dynamic memories in which access time increases logarithmically with the size of memory. In particular, we embed an interconnection pattern called the perfect shuffle into the memory. The results reported are directly applicable to MOS and magnetic bubble memories, but, unfortunately, cannot be applied to magnetic drums and disks.

In section II of this paper we derive the lower bound on access time that can be achieved in dynamic memories with enhanced interconnections. In section III, we describe a dynamic memory which actually meets this bound. By modifying the control of this memory, it can also be used in a search mode with an efficiency that rivals the efficiency of random access memories. The search mode of operation is discussed in Section IV. Another type of shift register, which is described in Section V, also makes use of the perfect shuffle, and is of importance for Fast Fourier Transform computations and for certain two-dimensional matrix computations.

II. A lower bound on minimum access time

The model of a dynamic memory that we adopt is one in which the data in memory is permuted in one of r different ways at each clock time. Thus each word in memory is connected to up to r other words.

One word in memory is distinguished as an input-output port. All data transfers between memory and the external world must go through the input-output word. In order to access a specific item, we must find where the item is in memory, and route it to the input-output port by a sequence of moves along the r interconnection patterns.

When $r=1$, the cyclic interconnection pattern is the only one that is suitable for a dynamic memory because it is the only permutation that places every word in memory on a path to the input-output word. Consequently, for $r = 1$, worst case access time is $N-1$, and average access time is $(N-1)/2$, where N is the size of memory. (For average access time we make the usual assumptions of uniform and independent distribution of accesses.)

When $r \geq 2$, access time can grow as the base r logarithm of N instead of linearly in N, as we show in the lemma below.

Lemma: A lower bound on the minimum worst case access time for a dynamic memory with r interconnection patterns is $M(r,N)$ where $M(r,N)$ is the smallest integer that satisfies

$$\frac{r^{M(r,N)+1} - 1}{r - 1} \geq N$$

Proof: At each clock time, we choose between 1 of r paths, so that the number of words that can be accessed in M clock times cannot exceed the number or r-ary sequences of length M or less. But this number is $\sum_{i=0}^{M} r^i = (r^{M+1} - 1)/(r-1)$ which proves the lemma.

As N becomes large, $M(r,N)$ grows as $\log_r (r-1) + \log_r N \cong \log_r N$. The bound is tabulated for various values of r and N in the Table I. The table has been constructed using the assumption that one item is available in zero cycles.

In the following section we present a pair of permutations that achieves minimum access time for $r = 2$, and N of the form $N = 2^m$. Note that $M(2,N) = \lceil \log_2 (N+1) - 1 \rceil = \log_2 N$ in this case.

TABLE I

N	4	8	16	32	64	128	256	512	1024
r = 2	2	3	4	5	6	7	8	9	10
3	1	2	3	3	4	5	6	6	6
4	1	2	2	3	3	4	4	5	5

M (r,N)

III. The Perfect Shuffle Shift Register

The perfect shuffle permutation pattern is shown in Fig. 1. The pattern takes its name from the fact that it is analogous to shuffling cards in a deck of playing cards. The memory cells on the left of Fig. 1 are shuffled by dividing the cells into two groups with a "cut" in the middle of the memory. The memory cells on the right receive data from the memory cells on the left by interlacing data from the two groups just as the two halves of a deck of playing cards are interlaced when undergoing a perfect shuffle.

Several properties of the perfect shuffle interconnection pattern have appeared in earlier papers, and are summarized below. (Pease, 1968; Batcher, 1968; Stone, 1971). For a derivation of the properties see Stone (1971).

Let $N = 2^m$, $m > 1$. and let the integers $0,1,\dots,N-1$ be stored in a memory of the type shown in Fig. 2. Then the memory has the following properties.

Property 1: A perfect shuffle moves the integer i to cell S(i) where S(i) is given by

$$\begin{aligned} S(x) &= 2x \quad \text{for } 0 \le x \le N/2 -1 \\ &= 2x-N+1 \quad \text{for } N/2 \le x \le N-1 \end{aligned}$$

Property 1': If the binary representation of i is

$$i = i_0 + i_1 \cdot 2 + \dots + i_{m-1} 2^{m-1}$$

then S(i) has the binary representation

$$S(i) = i_{m-1} + i_0 \cdot 2 + i_1 \cdot 2^2 + \dots + i_{m-2} \cdot 2^{m-1}.$$

Property 2: Consider the N/2 adjacent pairs of cells in this memory, as shown in Fig. 3. When the integers 0,1,... N-1 are placed in memory in ascending order, then the integers that are paired in adjacent cells differ only in the coefficient of 2^o in their binary expansions. After one shuffle, paired integers differ only in the coefficient of 2^{m-1} in their binary representations. After j shuffles, the paired integers differ only in the coefficient of 2^{m-j} in their binary expansions, $1 \leq j \leq m$. Consequently, after any number of shuffles, paired integers differ by precisely one coefficient in their binary expansions.

We shall make use of these properties of the perfect shuffle to construct a dynamic shift register memory containing N words in which every word can be accessed in no more than $\log_2 N$ clock times.

A diagram of the shift register memory for N = 8 appears in Fig. 4. Each word is connected to two other words in the register. The solid lines show the perfect shuffle connections. The dotted lines connect words in a pattern which is related to the perfect shuffle and is called the exchange-shuffle. Fig. 5a shows the exchange-shuffle interconnection pattern in isolation. The relationship to the perfect shuffle is made clear by Fig. 5b where we see that the exchange-shuffle is the permutation obtained by exchanging adjacent even-odd pairs of items, then shuffling them. By assumption, in Fig. 4 when a control signal EXCHANGE-SHUFFLE is applied, the data is shifted along the exchange-shuffle interconnections.

The three small registers shown in Fig. 4, are part of the addressing circuitry for the memory. Each contains three bits in the figure, and, in general, they contain $\log_2 N$ bits for dynamic memories with N words.

The A register is an address register that holds the address of the word to be accessed. The S register contains the address of the word that is currently in Word 0 of the memory, and the C register is a circulating shift register that always contains a single 1 bit. The S and C registers, together, describe precisely how the words are permuted in memory at any given instant. The memory data register that interfaces to the outside world is the word labeled "0".

The memory access mechanism is based upon the following notions. The S register holds the address of the item that is currently in Word 0. However, the S register alone does not describe the current permutation of the words in memory because up to $m = \log_2 N$ perfect shuffles can be applied to the memory, each resulting in a distinct permutation of the data in memory, while each shuffle leaves the contents of Word 0 fixed. After any number of shuffles the addresses of items in all even-odd pairs differ by precisely one coefficient in their binary expansion. Moreover, the coefficient is the same coefficient for all pairs. We call this coefficient the <u>pivot bit</u>. From the properties of the perfect shuffle, we see that the action of a shuffle causes the pivot bit to move cyclically in the sequence $2^{m-1}, 2^{m-2}, \ldots, 2^1, 2^0, 2^{m-1}$, etc. The C register contains a 1 in the pivot bit position, and 0's elsewhere. After each shuffle, the C register is updated by cyclically shifting the 1 to a position of less significance as shown in Fig. 4. The S register is not altered after a shuffle, since the contents of Word 0 are unchanged by a shuffle.

The actions that occur when the exchange-shuffle interconnections are used are best described by assuming that the exchange-shuffle is actually a combination of two distinct shifts as shown in Fig. 5b. Thus we shall consider only what happens when the data is permuted by the exchange operation shown in Fig. 5b, since we already know what happens when a shuffle occurs.

Since an exchange modifies the contents of Word 0, the S register must be updated after an exchange. Note that the pivot bit is not affected by an exchange, so that no change is made to the C register after an exchange. Note also that the pivot bit indicates exactly where the addresses associated with Word 0 and Word 1 differ so that the updating action for S register after an exchange is the following.

$$S \leftarrow S \oplus C$$

(The operator "$\oplus$" is the EXCLUSIVE OR operator)

At this point the addressing algorithm should be evident. To access an item, it has to be placed in Word 0, and thus its address must appear in the S register. If the address is already in the S register then the item is immediately available. If the address is not already in the S register, then we can place it there by complementing appropriate bits in the S register. But the S register is modified in the pivot bit position by an exchange, and the pivot bit position is altered by a perfect shuffle. The following algorithm in an ALGOL-like language shows how to generate a sequence of shuffles and exchange-shuffles to access any item from any given state of the memory.

```
LOOP:  if A = S then go to DONE;

       if $A \wedge C \neq S \wedge C$ then

          begin comment the addresses differ in the pivot
          bit position.  An EXCHANGE brings them into agreement;

             $S \leftarrow S \oplus C$;

             EXCHANGE-SHUFFLE;

          end

       else

          begin comment the addresses agree in the pivot bit
          position so that only a SHUFFLE is needed;

             SHUFFLE;

          end;

       comment a SHUFFLE is performed on both branches of the
       conditional statement above.  Hence, the C register must
       be updated, and the pivot bit is moved;


       CYCLE(C);

       go to LOOP

DONE:  Comment  the item with address A is now in Word 0;
```

Figure 6 shows two examples of the access algorithm. In the examples, the memory contains the integer i in the i^{th} address to enable the reader to observe the permutations of the addresses that occur during access.

The access algorithm clearly requires no more than $\log_2 N$ cycles because the S register contains $\log_2 N$ bits, and the processing of each bit takes exactly one cycle. In some cases, accesses can be done in fewer than $\log_2 N$ cycles, but this does not appear to have a significant effect on average access time. If we assume that address accesses are independent and identically distributed with a uniform distribution, then half of the accesses will require $m = \log_2 N$ cycles because in half of the cases the m^{th} pivot bit of the address will disagree with the m^{th} pivot bit of the S register. By similar reasoning, m-1 are required a quarter of the time, m-2 an eighth of the time, etc. Thus, $\overline{T}$, the average access time is given by

$$T = \sum_{i=0}^{m} 2^{-(i+1)}(m-i) = m \sum_{i=0}^{m} 2^{-(i+1)} - \sum_{i=0}^{m} i \cdot 2^{-(i+1)}$$

$$= m(1-2^{-(m+1)})-[1-2^{-m}(1+m/2)]$$

$$= m-1+2^{-m}$$

$$\cong m-1 = (\log_2 N)-1$$

We obtain an average access time of $\log_2 N$ when accesses are made to the addresses in numerical order since successive addresses then always differ in the least significant bit.

IV. Efficient Searching of a Shift Register Memory

The memory in Fig. 4 can be used for searching as well as for random access addressing, with the access times for the two modes differing only by a small constant factor. The search algorithm is nothing more than an adaptation of the familiar binary search algorithm. (Cf. Gear, 1969).

Assume that the items in memory are sorted in ascending order on their search key. When a search begins, the item may be at any address in the interval bounded by 0 and N-1. The first probe is made at the half-way point in this interval. Since N is even, the address may be either N/2 or $(N/2)^{-1}$, but we find it convenient to select N/2 as the probe address. (In general, we always break ties by selecting the even address.) If the item is at the probe address, the search terminates. If not, and the probe address contains an item with a lower key than the one for which we are searching, then our search can be limited to the interval bounded by $(N/2)^{-1}$ and N-1. If the probe finds a higher key, then the search interval is bounded by 0 and $(N/2)^{-1}$. In either case we probe at an address at the mid-point of the interval. The probing process is on successively smaller intervals repeated until either the search terminates successfully or the search fails on an interval of size 1.

To operate the memory of Fig. 4 in a search mode, we assume that access is made to Word 1 rather than to Word 0. The memory is initially placed in the state in which the address associated with Word 0 is the 0 address, and the pivot bit is the most significant bit in the C register. In this state, the address associated with Word 1 is N/2, which is the address of the first probe. If the search does not terminate after the first probe, then the second probe must be made at address N/2 + N/4 or at address N/4 depending on whether the search key is respectively greater than or less than the probe key. In the former case, the next item to probe is placed in Word 1 by the exchange-shuffle permutation, whereas in the latter case the shuffle permutation properly sets up the second probe. After the first probe and each successive probe the effect of a shuffle is to move the pivot bit to a less significant position, and thereby cause probes to occur at addresses that are successively N/2, N/4, N/8, ... higher than the address in the S register. Thus we obtain a succession of intervals of decreasing size. The exchange forces the succeeding probe to occur in the upper half of an interval instead of in the lower half of an interval because it causes the S register to increase by half of a probe interval.

The complete search algorithm is given below.

```
        comment we assume that the register named KEY contains the search
        key, that S contains 0, and that C contains a 1 in its high order
        bit position;
LOOP:   if KEY = WORD [1] then go to FOUNDIT;
        if C = 1 then go to NOTFOUND;
        if KEY>WORD [1] then
          begin comment increase the S register to force the next probe
                to occur in the upper half of the search interval. This
                requires an EXCHANGE;
                S ← S ⊕ C;
                EXCHANGE-SHUFFLE;
          end
```

```
else
   begin comment the next probe is to be in the lower half of the
         search interval.
         SHUFFLE;
   end;
CYCLE(C);
go to LOOP;
FOUNDIT:  comment the search was successful.
          The item is at address S ⊕ C;
          ...
NOTFOUND: comment at this point the search was not successful;
```

When the memory is in the proper initial state at most $\log_2 N$ probes are required to search a memory with N words. To place the memory in the proper initial state we must force a 0 into the S register, then place the pivot bit in the proper position. In the worst case, $\log_2 N$ cycles are required to initialize the S register, and, after initializing the S register, the pivot bit may have to be moved an additional $(\log_2 N) - 1$ places. Hence, the worst case access time in search mode is $3(\log_2 N) - 1$. Under the usual assumption of uniform distribution of search keys, the average time for a successful search after initialization is approximately $m-1 = (\log_2 N) - 1$ cycles. (This computation is the same as that outlined in the previous section.) We have previously shown that the average time required to place the S register in a specified state is m-1 cycles. After initialization of the S register, the pivot bit may be in any of m positions. If these are uniformly distributed, then $(m-1)/2$ cycles are required to initialize the C register. Thus on the average $5(m-1)/2$ cycles are required to access an arbitrary item in search mode.

We have left an important question unanswered with respect to efficient search operation of the memory. In search mode we require that the items be ordered in memory. If all of the items are sorted in a batch, then they can be sorted in approximately $k\ N \log_2 N$ cycles of a random access memory where k is a small constant. Although it is still an open question if sorting can be done in a shift register memory in a time proportional to $N \log_2 N$, it is quite clear that no more than $k\ N (\log_2 N)^2$ cycles are required to sort since each cycle of a random access memory can be simulated by $\log_2 N$ or fewer cycles of a shift register memory. In a conventional cyclic dynamic memory, sorting requires kN^2 cycles which is substantially less efficient than the memory we have described.

It is frequently the case that searching operations are interspersed with insertions and deletions of data in memory. In such a case, sorting after each insertion and deletion is much less efficient than other methods. In a dynamic memory, a crude method for inserting and deleting without sorting is to embed a cyclic interconnection pattern in memory, and perform the insertion or deletion during a sequence of N cyclic shifts of the memory.

More elegant methods for inserting and deleting are based on methods that require times proportional to $\log_2 N$ when performed in a random access memory. These methods are based upon inserting items as leaves of trees. In particular, the AVL tree algorithm is directly adaptable to our problem. [Adel'son - Vel'skiĭ and Landis, 1962; Foster, 1965]. Since it requires a time proportional to $\log_2 N$ for both insertion and deletion in a random access memory, the time cannot exceed $k(\log_2 N)^2$ for some constant k in the shift register memory.

In summary, it appears to be feasible to operate a shift register memory efficiently in search mode, and somewhat less efficiently for insertions, deletions, and sorting. It is rather interesting that searching can be done in a time that is within a constant factor of the binary search time in a random access memory. When we view the problem in a larger context and include the overhead of sorting, insertion, and deletion, we see that access constraints do materially affect the efficiency of the memory. The questions of efficiency are still unresolved however, in that there may exist sorting, insertion, and deletion algorithms that are better than those proposed here. In any case, for sufficiently large N, the shift register memory is substantially more effective for searching than a cyclically organized memory.

V. Other applications of the perfect shuffle interconnection pattern.

The memory described in the previous sections quite clearly has good characteristics that suggest it is of practical importance. However, it is not the only candidate for implementation, and it is of value to consider other interconnection patterns. In this section we investigate a memory which makes a perfect shuffle and a cyclic interconnection pattern as shown in Fig. 7. This memory is more effective for the applications mentioned below than is the memory described in previous sections. The applications are adaptations of algorithms for parallel processors (Stone, 1971).

The first application of the perfect shuffle is in a Fast Fourier Transform algorithm that is due to Pease (1968). A cursory examination of the FFT algorithm shows that in the first pass of the data, the items combined have indices that differ m-1 in the coefficient of 2^{m-1} in their binary expansions. On the second pass, the indices differ in the coefficient of 2^{m-2}, and the ith pass they differ in the coefficient of 2^{m-i}. Fig. 8 shows the pairs of items that are combined for $N = 8$. We see the familiar behavior of a shift pivot bit, and note that we can pair the appropriate items with a perfect shuffle. To perform the Fast Fourier Transform, we use the perfect shuffle once between passes. During each pass, pairs of items in even-odd pairs of words are using operands, producing results for the same pair of words. The entire memory is cycled during each pass. After $\log_2 N$ passes, the Fourier Transform has been computed in memory, but the items are scrambled in what is known as reverse binary order. Reordering the items is somewhat of a problem, and to do this efficiently we require some other mechanism. For special purpose FFT processors, it appears to be advantageous to embed an interconnection in memory for doing the reordering.

Another application suggested in Stone (1971) is that of taking a transpose of a matrix. If A is a matrix of dimension $2^s \times 2^t$ where the size of memory $N = 2^{st}$, then A can be changed from row major ordering to column major ordering in shuffles, and from column major to row major in t shuffles. Therefore, it is possible to access the elements of A sequentially both by rows and by columns in a shift register memory at the cost of a small overhead in time. Matrices with dimensions that are not powers of 2 can be transposed by storing them as upper left submatrices of matrices that do have dimensions that are power of 2. A dynamic memory that uses only the cyclic interconnection pattern is extremely inefficient for matrix processing when matrices have to be assessed both by row and by columns. To access a matrix of size $2^s \times 2^t$ by columns when it is stored by rows requires 2^t complete circulations of the memory since one column can be accessed during each complete circulation. Thus, for the matrix transpose, t shuffles and N complete circulations of the shift register memory in Fig. 7 do the job of $2^t \cdot N$ circulations of a conventional cyclic dynamic memory.

VI. Summary and conclusions

The results in Section II show the tradeoff between complexity of interconnections and access time in a dynamic memory. The value r=2 appears to have favorable characteristics for small memories. We have shown one way of achieving the best possible access time with two interconnection patterns, but undoubtedly there are many pairs of permutations that can perform equally well. A problem that remains unanswered at this time concerns the existence of permutations that achieve the access time bounds, but have desirable characteristics that the perfect shuffle memory does not have.

In particular, an important consideration for MOS shift register memories is the planarity of the interconnection pattern. For technologies in which interconnections should be planar, or cross-overs should be infrequent, there may be some difficulty in implementing the shift register connections described here. Obviously, the cyclic interconnection is a planar pattern, and therefore is advantageous for such technologies.

The results in this paper also have relevance to the problem of data communication in parallel processors. If each word in a dynamic memory corresponds to a processor in a parallel processor, then the interconnection patterns given here have the property that a particular datum can be transferred from any processor to a specified processor in minimum time.

In Section V, we briefly mentioned a memory in which the perfect shuffle and the cyclic patterns are combined. This pair of interconnections is extremely interesting to study, and is as yet incompletely understood.

Apparently, combinations of these two permutations can place the memory in more states than are reachable using the pair of interconnections of Section III. This could be very advantageous for some applications. The utility of the pair of interconnections would be significantly enhanced if an efficient access method were discovered.

References

1. Adel'son-Vel'skii and Landis, 1962. An algorithm for the organization of information, Doklady Akad. Nauk, Mathematics, Vol. 146, pp. 263-366.

2. Batcher, K. E., 1968. "Sorting networks and their application," 1968 Spring Joint Comp. Conf., AFIPS, Proc., Vol. 32, Washington, D.C.: Thompson, pp 307-314.

3. Foster, C. C., 1965. Proc. ACM Nat'l Conf., pp. 192-205.

4. Gear, C. William, 1969. Computer Organization and Programming, McGraw-Hill, 1969.

5. Pease, M., 1968. An adaptation of the Fast Fourier Transform for parallel processing. JACM, Vol. 15, No. 2, pp. 252-264.

6. Stone, H. S., 1971. Parallel processing with the perfect shuffle. To appear in IEEE Computer Transactions.

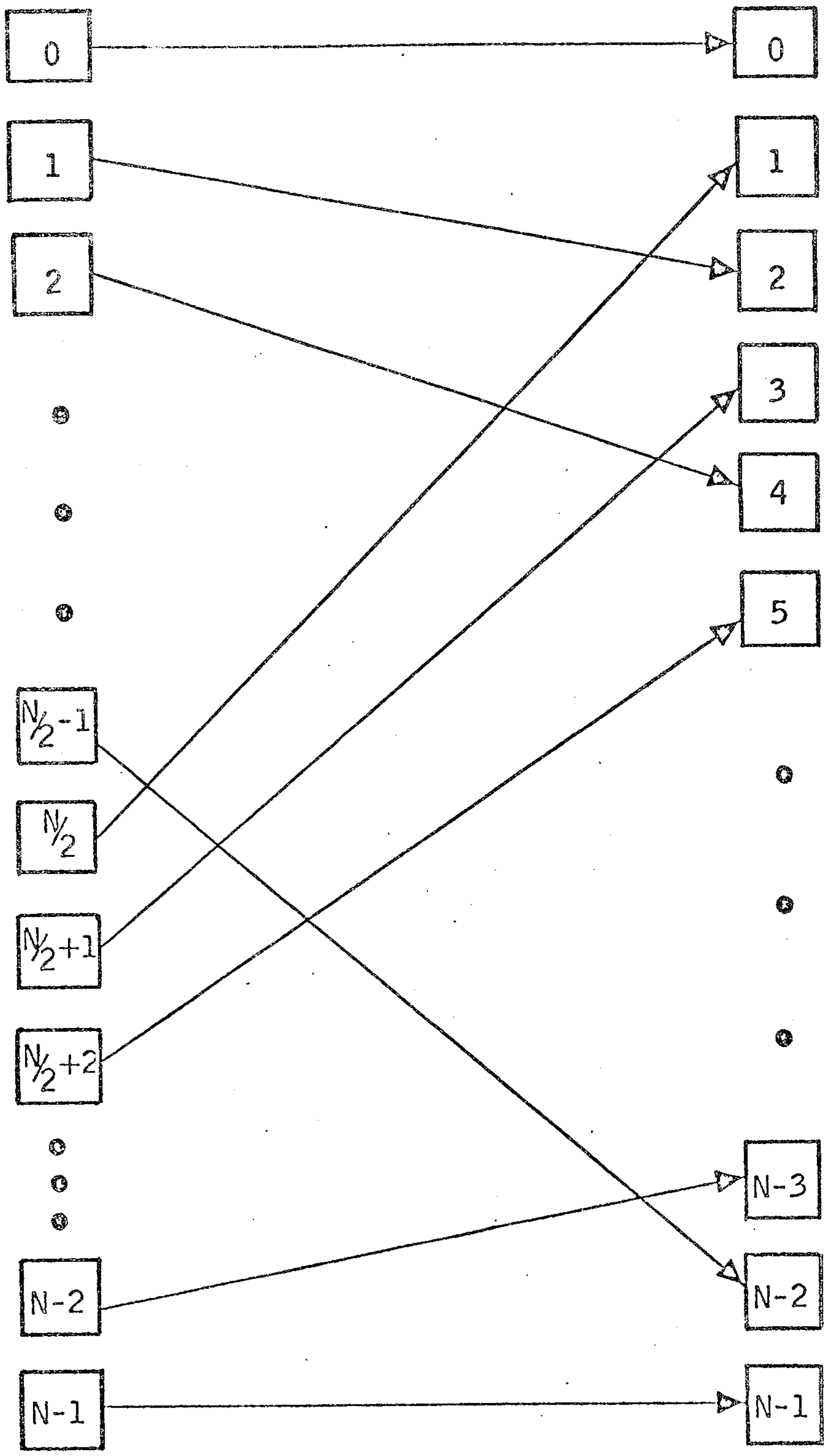


Figure 1. The perfect shuffle of an N-element vector.

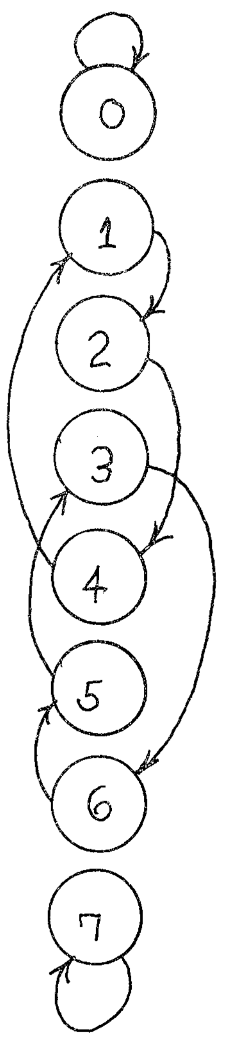


Fig. 2. A dynamic memory with the perfect shuffle interconnection pattern. N = 8.

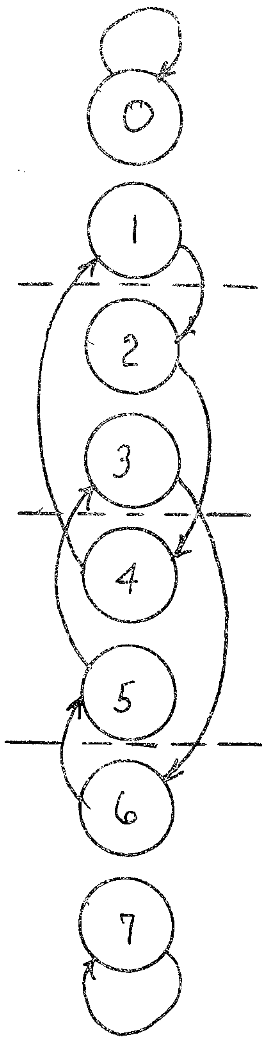


Fig. 3. Adjacent pairs of cells in a perfect shuffle interconnection pattern. N = 8.

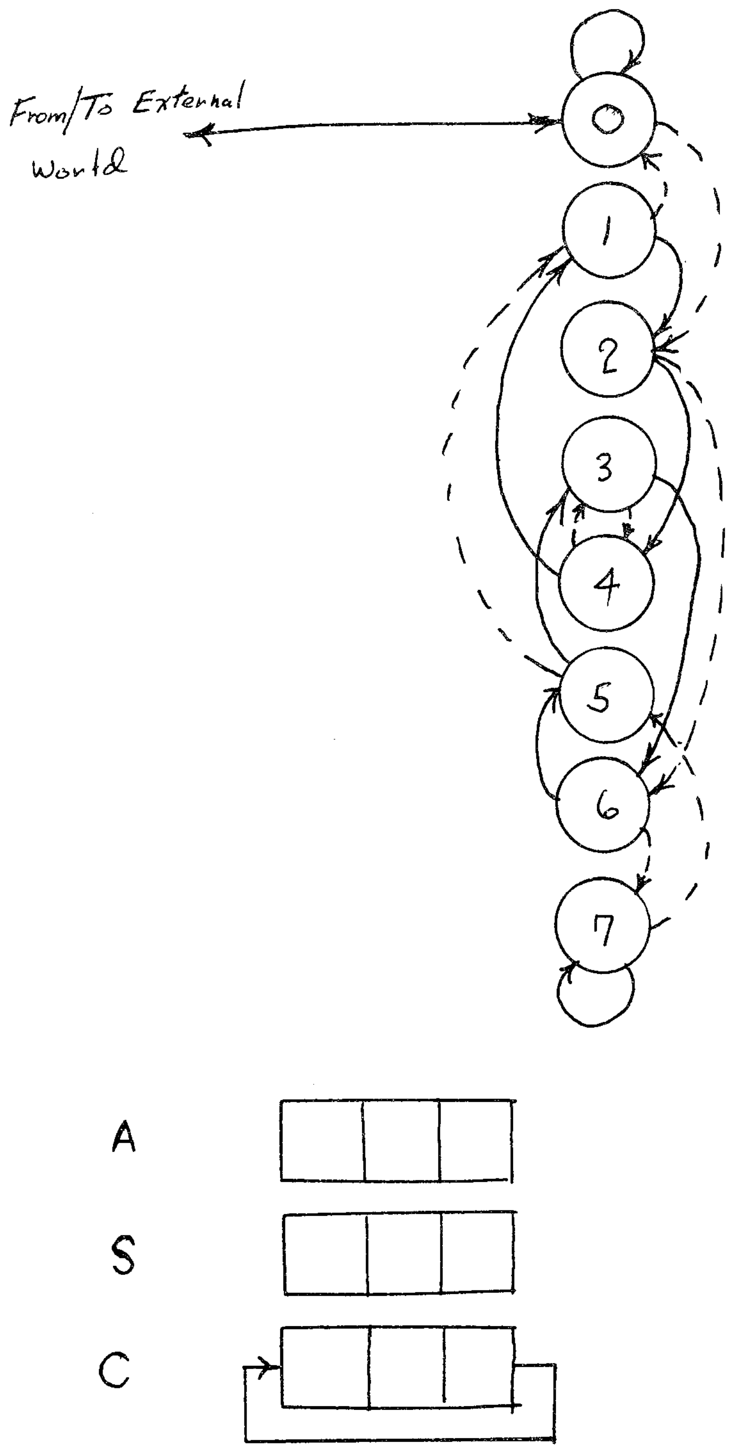
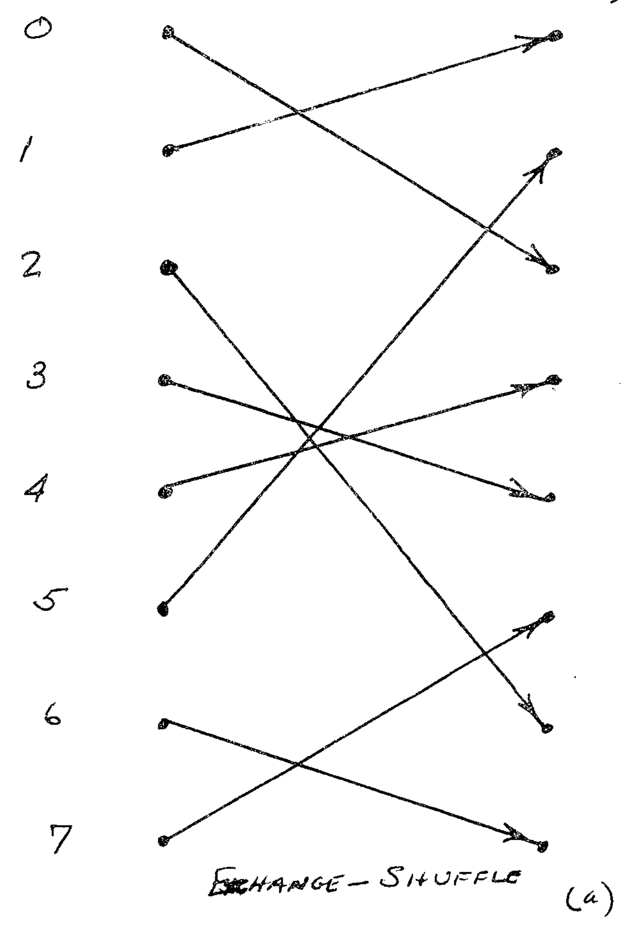


Fig. 4. The perfect shuffle shift register memory.



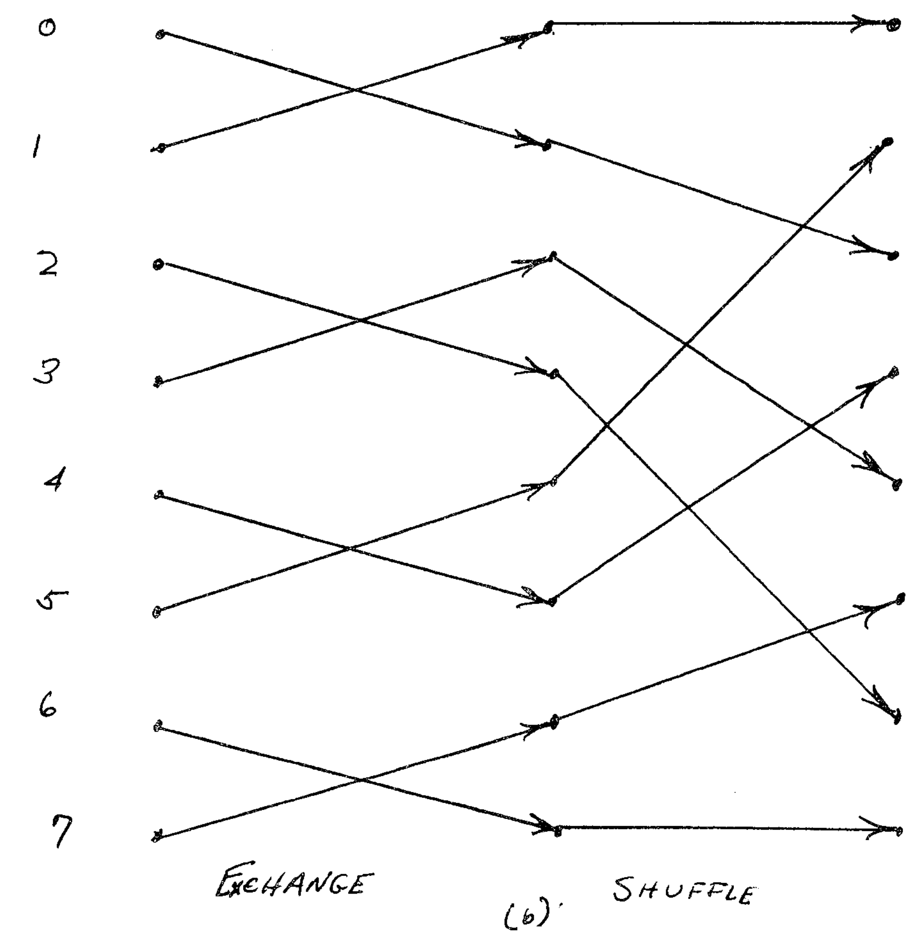


Fig. 5. (a) The Exchange-shuffle. (b) The structure of the Exchange-shuffle. N = 8.

Memory Contents

Word				
0	0	4	4	5
1	4	6	5	1
2	1	0	6	4
3	5	2	7	0
4	2	5	0	7
5	6	7	1	3
6	3	1	2	6
7	7	3	3	2
A	101	101	101	101
S	000	100	100	101
C	100	010	001	100
Actions:	Exchange Shuffle	Shuffle	Exchange-Shuffle	

0	5	1	3
1	1	3	2
2	4	5	1
3	0	7	0
4	7	0	7
5	3	2	6
6	6	4	5
7	2	6	4
A	011	011	011
S	101	001	011
C	100	010	001
Action:	Exchange Shuffle	Exchange Shuffle	

Fig. 6. Examples of memory access in a perfect shuffle shift register memory.

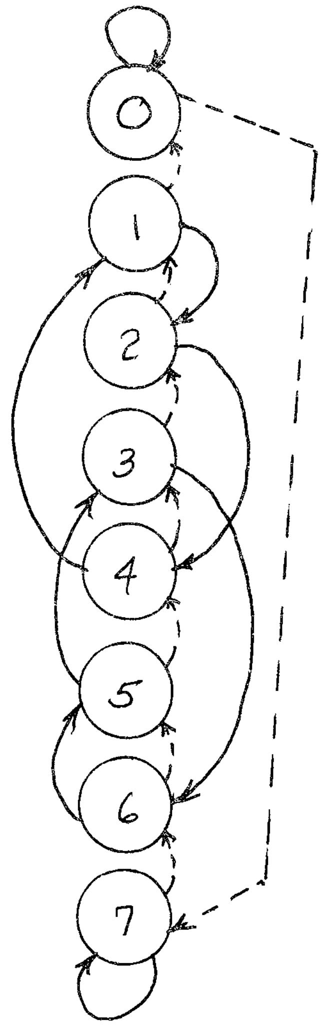


Fig. 7. A shift register memory which uses the perfect shuffle and the cyclic shift interconnections. N = 8.

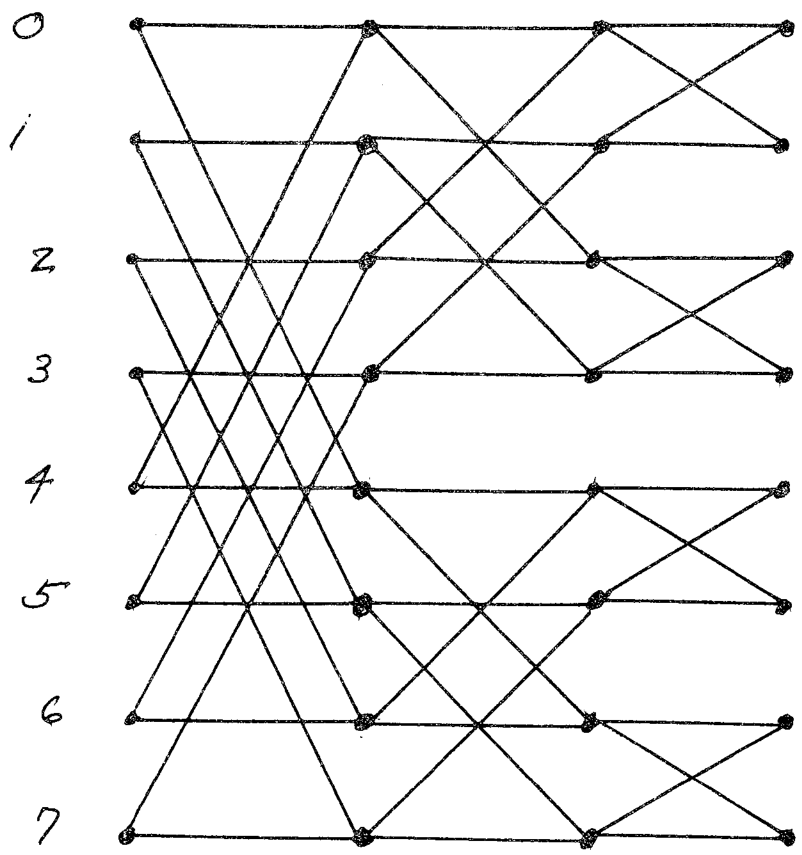


Fig. 8. Data flow in a Fast Fourier Transform computation. N = 8.

JOINT SERVICES ELECTRONICS PROGRAM
REPORTS DISTRIBUTION LIST

Department of Defense

No. of Copies	
1	Dr. Edward M. Reilley Asst. Director (Research) Office of Director of Defense Research & Engrg Room 3C128, The Pentagon Washington, D.C. 20301
1	Director for Materials Sciences Advanced Res. Projects Agency 1400 Wilson Blvd. Arlington, Virginia 22209
1	Chief, R&D Division (340) Defense Communications Agency Washington, D.C. 20301
12	Defense Documentation Center Attn: DDC-TCA Cameron Station Alexandria, Virginia 22314
1	LTC Norman D. Jorstad Weapons Systems Evaluation Group 400 Army-Navy Drive Arlington, Virginia 22202
1	Dr. A. D. Schnitzler Institute for Defense Analyses Science and Technology Division 400 Army-Navy Drive Arlington, Virginia 22202
1	Central Intelligence Agency Attn: CRS/AD/Publications Washington, D.C. 20505

Department of the Air Force

No. of Copies	
1	Headquarters/USAF (AF/RDPE) Washington, D.C. 20330
1	Headquarters USAF/RDPS Washington, D.C. 20330
1	Headquarters USAF/RDSD The Pentagon Washington, D.C. 20330
1	Colonel E. P. Gaines, Jr. ESD (MCD) L. G. Hanscom Field Bedford, Massachusetts 01730
5	Dr. Lloyd A. Wood, Director Electronics & Solid State Sci. Air Force Office of Sci. Res. 1400 Wilson Blvd. Arlington, Virginia 22209
1	Rome Air Development Center Attn: Documents Library (TDLD) Griffiss AFB, New York 13440
1	Mr. H. E. Webb, Jr. (ISCP) Rome Air Development Center Griffiss AFB, New York 13440
1	AFSC (CCJ/Mr. Irving R. Mirman) Andrews AFB Washington, D.C. 20331
1	Dr. L. M. Hollingsworth AFCRL (CA) L. G. Hanscom Field Bedford, Massachusetts 01730
2	Headquarters ESD (TRI) L. G. Hanscom Field Bedford, Massachusetts 01730
1	Professor R. E. Fontana, Head Dept. of Electrical Engineering Air Force Institute of Tech. Wright-Patterson AFB, Ohio 45433
1	Dr. H. V. Noble, AFAL/TE Chief, Electronics Tech. Division Air Force Avionics Laboratory Wright-Patterson AFB, Ohio 45433

No. of Copies	
1	Director U.S. Army Advanced Materiel Concepts Agency 2461 Eisenhower Avenue Alexandria, Virginia 22314
1	Director Walter Reed Army Inst. of Res. Walter Reed Medical Center Washington, D.C. 20012
1	Mr. H. T. Darracott (AMXAM-FT) U.S. Army Advanced Materiel Concepts Agency 2461 Eisenhower Avenue Alexandria, Virginia 22314
1	Commanding Officer (AMXRD-BAD) U.S. Army Ballistics Res. Lab. Aberdeen Proving Ground Aberdeen, Maryland 21005
1	U.S. Army Munitions Command Attn: Science & Technology Inf. Branch, Bldg 59 Picatinny Arsenal, SMUPA-RT-S Dover, New Jersey 07801
1	Dr. Herman Robl Deputy Chief Scientist U.S. Army Research Office (Durham) Box CM, Duke Station Durham, North Carolina 27706
1	Richard O. Ulsh (CRDARD-IP) U.S. Army Research Office (Durham) Box CM, Duke Station Durham, North Carolina 27706
1	Technical Director (SMUFA-A2000-107-1) Frankford Arsenal Philadelphia, Pennsylvania 19137
1	Redstone Scientific Inf. Center Attn: Chief, Document Section U.S. Army Missile Command Redstone Arsenal, Alabama 35809

No. of Copies	
1	Commanding General U.S. Army Missile Command Attn: AMSMI-RR Redstone Arsenal, Alabama 35809
1	Commanding General U.S. Army Strategic Communications Command Attn: SCC-ATS (Mr. Peter B. Pichetto) Fort Huachuca, Arizona 85613
1	Dr. Homer F. Priest Chief, Materials Sci. Division Building 292 Army Materials & Mechanics Res. Ctr. Watertown, Massachusetts 02172
1	Dr. Berthold Altmann (AMXDO-TI) Harry Diamond Laboratories Connecticut Ave. & Van Ness St. N.W. Washington, D.C. 20438
1	Commanding General USACDC Institute of Land Combat Attn: Technical Library, Rm 636 2461 Eisenhower Avenue Alexandria, Virginia 22314
1	Commandant U.S. Army Air Defense School Attn: Missile Sci. Div. C&S Dept. P.O. Box 9390 Fort Bliss, Texas 79916
1	Commandant U.S. Army Command & General Staff College Attn: Acquisitions, Lib. Div. Fort Leavenworth, Kansas 66027
1	Dr. H. K. Ziegler (AMSEL-KL-D) Army Member, TAC/JSEP U.S. Army Electronics Command Fort Monmouth, New Jersey 07703

Department of the Navy

No. of Copies	
3	Director, Electronic Programs Attn: Code 427 Office of Naval Research 800 North Quincy Street Arlington, Virginia 22217
1	Mr. Gordon D. Goldstein, Code 437 Information Systems Program Office of Naval Research 800 North Quincy Street Arlington, Virginia 22217
1	Commander Naval Security Group Command Naval Security Group Headquarters Attn: Technical Library (G43) 3801 Nebraska Avenue, N.W. Washington, D. C. 20390
1	Director Naval Research Laboratory Attn: Mr. A. Brodzinsky, Supt. Electronics Div. Washington, D. C. 20390
1	Director Naval Research Laboratory Attn: Code 5200 Washington, D. C. 20390
1	Dr. Herbert Rabin, Code 7000 Assoc. Dir. of Res. for Space & Tech., Actg. Naval Research Laboratory Washington, D. C. 20390
2	Director Naval Research Laboratory Attn: Library, Code 2629 (ONRL) Washington, D. C. 20390
1	Dr. G. M. R. Winkler Director, Time Service Division U. S. Naval Observatory Washington, D. C. 20390
1	Naval Air Systems Command AIR-310 Research Administrator Room 424, JP-1 Washington, D. C. 20360
1	Naval Ship Systems Command Ship 035 Washington, D. C. 20360
1	Dr. A. L. Slafkosky Scientific Advisor Commandant of the Marine Corps (Code AX) Washington, D. C. 20380
1	U. S. Naval Weapons Laboratory Dahlgren, Virginia 22448
1	Naval Electronic Systems Command Attn: Code 0311, Rm. 7W12, NC #1 Department of the Navy Washington, D. C. 20360
2	Commander U. S. Naval Ordnance Laboratory Attn: Librarian White Oak, Maryland 20910
1	Director Office of Naval Research Boston Branch 495 Summer Street Boston, Massachusetts 02210
1	Commander (ADL) Naval Air Development Center Attn: NADC Library Johnsville, Warminster, Pennsylvania 18974
1	Commanding Officer Naval Missile Center Attn: 5632.2, Technical Library Point Mugu, California 93042
1	W. A. Eberspacher, Associate Head Systems Integration Division, Code 5340A U. S. Naval Missile Center Point Mugu, California 93041

No. of
Copies

1 Director
Coordinated Science Laboratory
University of Illinois
Urbana, Illinois 61801

1 Director
Stanford Electronics Laboratory
Stanford University
Stanford, California 94305

1 Director
Microwave Laboratory
Stanford University
Stanford, California 94305

1 Director
Electronics Research Laboratory
University of California
Berkeley, California 94720

1 Director
Electronics Sciences Laboratory
University of Southern California
Los Angeles, California 90007

1 Director
Electronics Research Center
The University of Texas at Austin
Engineering-Science Bldg. 110
Austin, Texas 78712

1 Director of Laboratories
Division of Engineering &
Applied Physics
Harvard University
Pierce Hall
Cambridge, Massachusetts 02138

1 Dr. G. J. Murphy
The Technological Institute
Northwestern University
Evanston, Illinois 60201

1 Dr. John C. Hancock, Head
School of Electrical Engineering
Purdue University
Lafayette, Indiana 47907

No. of
Copies

1 Dept. of Electrical Engineering
Texas Technological University
Lubbock, Texas 79409

1 Aerospace Corporation
P. O. Box 95085
Attn: Library Acquisitions Group
Los Angeles, California 90045

1 Airborne Instruments Laboratory
Deerpark, New York 11729

1 The University of Arizona
Dept. of Electrical Engineering
Tucson, Arizona 85721

1 Engineering & Mathematical
Sciences Library
University of California
at Los Angeles
405 Hilgred Avenue
Los Angeles, California 90024

1 Sciences-Engineering Library
University of California
Santa Barbara, California 93106

1 Professor Nicholas George
California Inst. of Technology
Pasadena, California 91109

1 Aeronautics Library
Graduate Aeronautical Laboratories
California Inst. of Technology
1201 E. California Blvd.
Pasadena, California 91109

1 Hunt Library
Carnegie-Melon University
Schenley Park
Pittsburgh, Pennsylvania 15213

1 Dr. A. G. Jordan
Head of Dept. of Electrical Engrg.
Carnegie-Mellon University
Schenley Park
Pittsburgh, Pennsylvania 15213