DYNAMIC MEMORY INTERCONNECTIONS

FOR RAPID ACCESS

by

Balakrishna R. Iyer and J. B. Sinclair

November 1980

Technical Report 8004
DYNAMIC MEMORY INTERCONNECTIONS FOR RAPID ACCESS

Abstract: Certain aspects of bulk storage technology development have required the study of dynamic memory interconnections. Various schemes to interconnect physical storage locations have been proposed in the literature. Taking into consideration the fact that dynamic memories are characterised by a small number of i/o ports, these schemes attempt to minimise random access time or reduce sequential access time. In this paper a scheme is proposed which combines concepts of interleaving and properties of the perfect shuffle interconnection network to permit random access of items in a large bulk store in less than logarithmic time and subsequent sequential access in unit time. Asymptotic behaviour of this scheme is also examined.

1 INTRODUCTION

Technological advances in recent years indicate that new bulk storage devices may soon compete directly with conventional disks and drums. Integrated devices, consisting of magnetic bubbles, charge coupled devices or MOS shift registers, are already in limited use.

Stone\(^3\) predicts the characteristics of future bulk memories, on the basis of present day experience. Each physical location will be limited to a small fan-out, so that the contents of any given location may be routed to one or more of a small number of locations. A memory module will also be characterised by relatively few i/o ports compared to the number of memory locations in a module. Dynamic storage may also
be anticipated in bulk memories of the future. Dynamic storage schemes require a distinction to be made between logical addresses and physical addresses of data items. Data elements will migrate in physical storage. At each time unit the correspondence between the logical addresses of data and physical addresses of locations where the data reside is specified by a storage mapping function, abbreviated as the s-mapping. The memory access mechanism will use the s-mapping to locate the required data element and bring it to an i/o port.

Ideally, it would be desirable to be able to access every location in unit time. This requires the implementation of an interconnection network among memory elements that would permit the contents of an arbitrary physical memory location to be transmitted directly to an i/o port. Clearly, a network that provides data paths between each memory location and the i/o port is a possibility. However, practical considerations of circuit complexity and cost motivate the implementation of interconnection patterns which provide only a few paths leading out of a physical memory location for information transfer. If, on the other hand, the fan-out of each location is restricted to 1, then the memory locations must be cyclically connected, since only a cyclic connection offers access to every location under this restriction. For such a memory of N locations, with a fixed number of i/o ports, random access would take an average of O(N) time.

Stone\textsuperscript{2} examines the situation in which there are only two paths leading out of each location. He shows that by using the memory interconnection scheme that he suggests, any random data item can be
accessed in \( \log N \) time where \( N \), the number of memory locations is a power of 2, and that this is a tight lower bound for random access under this restriction. However, subsequent sequential access may take \( \log N \) time for every individual data item. Aho and Ullman\(^1\) and Stone\(^4\) show that Stone’s scheme can be altered to allow random access of data items in worst case \( 2 \log N \) time and average case \( 1.5 \log N \) time. Thereafter sequential access is accomplished in unit time. In this organisation the number of memory locations interconnected is \( N - 1 \). Lenfant\(^5\) generalises Aho and Ullman’s scheme to cases where \( N \) is not a power of 2. He also proposes a scheme wherein random access takes an average of approximately \( 1.25 \log N \) time, for \( N = 2^k \) or \( 2^k - 1 \). Wong and Tang\(^6\) have proposed another scheme which reduces the average random access time to \( \log N - 1.333 \). However the worst case random access time for their scheme is \( 1.5 \log N - 1 \). Both these schemes permit sequential access in unit time after the initial random access. Wong and Tang’s memory is restricted to have \( N - 1 \) locations, where \( N \) is a power of 2.

Another approach to the problem has been indicated by Stone\(^4\). He suggests construction of a sequence of allowable permutations that brings the contents of each location in the network to the i/o port exactly once. Such a sequence is called a tour. Stone has shown the existence of tours for \( N = 4, 8, 16 \). Further study has been carried out by Morris et al\(^7\), showing the existence of tours for memories of sizes \( N = 2^r \), \( 2^r \leq N \leq 6 \). The permutations considered in the tour problems are those originally proposed by Stone for dynamic memory interconnection. No general theory exists to assure the existence of tours for all \( N \). Furthermore this approach requires address recoding which is likely to affect the scheme adversely in terms of both time and space.
In section II the s-mapping associated with Stone’s scheme will be characterised and an algorithm presented, based on this scheme, to implement a control unit allowing access to any logical address in \( O(\log N) \) time. Section III discusses a memory organisation which interleaves logical addresses among sets of memory locations. The interconnection network consists of data paths within each set of locations, with every location in a set having outputs connected to two other locations belonging to the same set. With one exception there is no connection across set boundaries. A single cell, not belonging to any of these sets serves as an i/o port and is connected to one distinguished cell in each set by a bidirectional data path. Access to a random logical address is then possible in less than logarithmic time, while also permitting subsequent sequential access in unit time. This organisation is shown to be practical for even large memories. The control mechanism to implement such a scheme is described in section IV. Section V discusses theoretical considerations of the above organisation. A modification to the proposed organisation is introduced to analyse the asymptotic behaviour of the access mechanism.

II STONE’S SHUFFLE AND SHUFFLE EXCHANGE INTERCONNECTION

The interconnection scheme proposed in this paper is based on Stone’s scheme for accessing elements in a dynamic memory. Addresses of the N elements are represented as bit strings. In order to optimise the utilisation of the address field, computer memories consist of a number of locations which is an integer power of 2. For this reason, Stone has
restricted his analysis to cases for which \( N = 2^m \), with \( m \) being a positive integer. The \( N \) elements are interconnected by two permutation networks. During each clock pulse one of the two permutations is selected and uniformly applied to the memory. This is done by gating the output of each location to one of the two locations to which it is connected. All locations in the memory are gated simultaneously.

The memory has only one i/o port for the \( N \) memory locations. Access to any logical address requires transfer of the contents of the logical address from its physical location at the time of the access request to the i/o port. This is done through repeated applications of the two permutations implemented in this scheme. At the application of a permutation, the data element in physical location \( i \) is routed to physical location \( d(i) \). The mapping \( d \) is a permutation function and will be referred to as the \( d \)-mapping. Stone's scheme implements two \( d \)-mappings, defined as follows:

a) The **Perfect Shuffle** The perfect shuffle interconnection logically is analogous to a shuffle of a pack of cards\(^2\). The perfect shuffle is defined for even values of \( N \) by the following \( d \)-mapping:

\[
\begin{align*}
    ds(i) &= 2i \quad \text{for } 0 \leq i \leq N/2 - 1 \\
    &= 2(i-N/2) + 1 \quad \text{for } N/2 \leq i \leq N - 1
\end{align*}
\]

[1]

The perfect shuffle interconnection is illustrated in Fig. 1 for \( N = 8 \). Each memory element has been drawn twice to illustrate the source and destination of an interconnection path. For purposes of
brevity the perfect shuffle will henceforth be referred to as the shuffle.

Stone\(^2\) has shown that if \(N = 2^m\) and \(i = \sum_{k=0}^{m-1} b_k \times 2^k\), where \(b_k\) is either 0 or 1 for \(0 \leq k < m\), then \(ds(i) = b_{m-1} \times 2^0 + \sum_{k=0}^{m-2} b_k \times 2^{k+1}\). A shuffle transformation of \(i\) is equivalent to a left cyclic shift of the binary representation of \(i\). This will be indicated by \(ds(i) = 2^p \% i\). Note that \(ds(ds(i))\) can be written as \(2^p(2^p i)\) which will be notationally equivalent to \((2^p \% i)\). Similarly \(ds(ds(\ldots(ds(i)\ldots)\ldots))\) the result of \(p\) such applications of the \(d\)-mapping will be written as \((2^p \% i)\).

B) The Shuffle Exchange The shuffle exchange interconnection can be considered as a composition of two subpermutations. The first subpermutation is a shuffle permutation. This is followed by an exchange permutation. Adjacent locations are defined as those locations whose physical addresses differ only in the least significant bit. An exchange consists of an interchange of data elements between adjacent locations. An element in physical location \(j\) is routed to physical location \(j^*\), where \(j^*\) is obtained by complementing the least significant bit of \(j\). The shuffle exchange is illustrated for \(N = 8\) in Fig. 2.

The \(s\)-mapping for Stone's proposed memory interconnection network can be characterised with the following notation. Let \(j\) be an integer such that \(0 \leq j < 2^m\). Then \(j^*[b_{m-1} b_{m-2} \ldots b_k \ldots b_1 b_0]\) is the \(m\)-bit binary integer obtained from the \(m\)-bit representation of \(j\) by
complementing the k’th bit of j if boolean variable \( b_k = 1 \), for \( 0 \leq k < m \).

A data item in a location with physical address \( j^{[b_{m-1} b_{m-2} \ldots b_2 b_1 b_0]} \) is routed by an exchange permutation to a location with physical address \( j^{[b_{m-1} b_{m-2} \ldots b_2 b_1 b_0*]} \), where \( b_0* \) is the complement of \( b_0 \).

Using this notation, the shuffle can be represented as

\[
ds(j^{[b_{m-1} b_{m-2} \ldots b_1 b_0]}) = 2 \$ (j^{[b_{m-1} b_{m-2} \ldots b_1 b_0]})
\]

\[= (2 \$ j)^{[b_{m-2} b_{m-3} \ldots b_0 b_{m-1}]}.
\]

A shuffle exchange is described by

\[
dx(j^{[b_{m-1} b_{m-2} \ldots b_1 b_0]}) = (2 \$ j)^{[b_{m-2} b_{m-3} \ldots b_0 b_{m-1}*]}.
\]

Assume that \( N \) data elements are initially placed in memory so that their logical addresses are equal to their physical addresses. In terms of the \( s \)-mapping this can be represented as \( s(i) = i = (2^0 \$ i)^{[00 \ldots 0]} \). Equations [3] and [4] show that after an arbitrary sequence of shuffles and shuffle exchanges, the physical address of the element with logical address \( i \) is given by the storage mapping function \( s(i) \), where \( s(i) \) is of the form

\[
(2^P \$ i)^{[b_{m-1} b_{m-2} \ldots b_1 b_0]}, \text{ for some combination of bits } b_k \text{ such that}
\]
0 \leq p < m, 0 \leq k < m and 0 \leq i < 2^m.

The location with physical address 0 is the i/o port. The control circuitry for data access is built around three registers - the A-register, the B-register and the C-register. When logical address \( j \) is to be accessed, the bit representation for \( j \) i.e. \( j_{m-1} j_{m-2} \ldots j_1 j_0 \) is parallel-loaded into the A-register. The physical address corresponding to logical address \( j \) is given by \((2^p \& j)[b_{m-1} b_{m-2} \ldots b_1 b_0]\) for some \( p \) and bit vector \( b \). To access logical address \( j \), the contents of the above physical address must be brought to physical address 0, the i/o port. The memory must be subjected to a sequence of shuffles and shuffle exchanges such that at the completion of this process the s-mapping, described by \( s(i) = (2^p \& i)[b_{m-1} b_{m-2} \ldots b_1 b_0] \), where \( s(j) = 0 \).

At all times the B-register contains the logical address of the data item in the i/o port. Hence if \( s(i) = (2^p)[b_{m-1} b_{m-2} \ldots b_1 b_0] \) the B-register contains the bit string \( b_{p-1} b_{p-2} \ldots b_0 b_{m-1} \ldots b_p \). The C-register contains the bit representation of \( 2^{m-1-p} \). The algorithm in Fig. 3 determines a sequence of shuffles and shuffle exchanges that will bring the contents of logical address \( j \) to the i/o port. At the end of the access sequence the s-transformation is described by \( s(i) = (2^p)[j_{m-p-1} \ldots j_0 j_{m-1} \ldots j_{m-p}] \).

The algorithm in Fig. 3 brings the contents of logical address \( j \) to physical location 0 in \( \log N \) time units.
III THE DECK INTERCONNECTION NETWORK

A interconnection network capable of implementing shuffle and shuffle exchange permutations, as described in section II, will be called an SSE network. An SSE module is a set of memory locations interconnected by an SSE network. The interconnection network proposed in this section will be referred to as the deck interconnection network. The deck interconnection network can be visualised as several SSE networks, each consisting of the shuffle and shuffle exchange interconnections, with an additional interconnection structure tied to the i/o ports which serves to integrate the SSE modules into a single memory unit.

The deck is characterised by two parameters: \( N \), the total number of memory locations; and \( r \), the number of distinct SSE networks embedded in the deck. The memory is partitioned into \( r \) equal sets and each set of \( N/r \) locations is interconnected by an SSE network. \( N/r \) is chosen to be a power of 2 to simplify the control mechanism. The SSE module discussed by Stone has a single i/o port. Now these "i/o ports" for the \( r \) SSE modules will be regarded as access windows. By performing a series of shuffles and shuffle exchanges within an SSE module, the contents of a physical location can be brought to an access window in \( \log(N/r) \) time. An additional location, the i/o port for the entire memory, is connected to each of these \( r \) access windows via an a \( r \)-to-1 multiplexer and a 1-to-\( r \) demultiplexer. A data item can hence be routed between any distinguished location and the i/o port. An example of a deck interconnection network for \( N = 32 \) and \( r = 4 \) is shown in Fig. 4.
It will be shown that values of $r$ for typical computer applications are small. Hence data can be routed from an access window to the i/o port in essentially constant time, and this time will be ignored in the rest of this section and in section IV. Access of consecutive locations from the same SSE module takes $\log(N/r)$ time units per access. If the deck network is to deliver one data element per time unit during sequential access, then logically consecutive data items must reside in different SSE modules. Hence logical addresses are interleaved among the $r$ SSE modules. If the SSE modules are labeled 0 through $r-1$, then the data element with logical address $i$ is stored in the SSE module $i \mod r$. Within the SSE module, the data item is treated as if it had logical address $i/r$. For a random access of logical address $j$, the control unit must be able to recognise that the contents of the logical address resides in the SSE module $j \mod r$, and perform the necessary sequence of shuffles and shuffle exchanges, one per time unit, to route the contents of the logical address into the access window.

Anticipating sequential access, the control unit starts routing the contents of the $(j+1)$th logical address in the $[(j+1) \mod r]$th SSE module to the distinguished location of that SSE module, this sequence lagging the sequence in the $j \mod r$th SSE module by one time unit. The $(j+2) \mod r$th SSE module is made to behave similarly with respect to the $(j+1) \mod r$th SSE module. This behaviour is repeated for cyclically adjacent SSE modules. If there are at least $\log(N/r)$ SSE modules, then after an initial delay of $\log(N/r)$ time units, data from consecutive logical addresses can be delivered to the i/o port at the rate of one item per time unit. Thus it is required that
\[ r \geq \log \left( \frac{N}{r} \right) . \]

Minimising \( r \) reduces the size of the multiplexer required between the access windows and the i/o port and minimises the number of independent control units required. In the best case \( r \) can be selected so that \( r = \log \left( \frac{N}{r} \right) \).

When \( r \) is chosen as above, the random access time \( \log N/r \) is bounded as follows:

\[ r = \log N/r = \log N - \log r \]

Hence \( r \leq \log N \)

Thus \( \log r \leq \log(\log N) \)

and \( \log N - \log r \geq \log N - \log(\log N) \).

This relation shows that for such a choice of \( r \), random access time is less than logarithmic and is bounded between \( \log N - \log(\log N) \) and \( \log N \).

### IV IMPLEMENTATION

A schematic for the deck interconnection network, for \( N = 2^{11} \) and \( r = 8 \) is shown in Fig. 5. The control paths are represented by dotted lines and data paths are represented by solid lines.

The main control unit (MCU) chooses the SSE module that contains the logical address to be accessed. Associated with each SSE module is some control circuitry (CC) capable of initiating a sequence
of shuffles and shuffle exchanges to access a logical address within the SSE module, as described in Section II. The MCU loads a particular CC with the logical address to be accessed and this triggers a sequence of shuffles and shuffle exchanges within the SSE module. The MCU operates on cyclically consecutive CCs. Log N/r time units after the sequence is initiated the required data is available at the distinguished cell of the SSE module. The CC then sends a signal to transfer the data to the i/o port. Arbitrary choice of r can make the operations |1/r| and i mod r cumbersome to implement. However, if r is chosen as a power of two, the two operations above reduce to mere bit selection, simplifying the control unit. Table 1. shows how N grows with r, where r and N are related by $r + \log r = \log N$.

It is easily observed from this table that even for a dynamic memory of 1 million logical addresses only 16 SSE networks are required. This justifies the earlier statement that r is small for even very large memories. It is desirable for large size memories to be implemented as several SSE networks rather than one. In this way, the problem of crossing connections on the integrated circuit that would implement the store is reduced. Fabrication of circuitry on the device is simplified due to the replication among the SSE networks constituting the deck.
V ASYMPTOTIC BEHAVIOUR

In section III, use of a r-to-l multiplexer and a l-to-r demultiplexer has been justified by the fact that r, the number of memory subsets, tends to be small in most dynamic memory implementations. This may not be true for extremely large memories. The implicit assumption that control information may be passed from the MCU to the CC of any SSE module in negligible time is also suspect in this case. Indeed, the propagation of control information is one issue that prevents the consideration of the binary tree as an interconnection structure. This section discusses the modifications needed to be made to the deck memory organisation in these cases. The only restriction made is to require N to be a power of 2.

As in section III the N memory locations are divided into r sets, each set interconnected by an SSE network. Criteria for the choice of r are discussed later in this section. It will turn out that r would have to be restricted to a power of 2. As before consecutive logical addresses are interleaved among the r memory subsets. Random access in the memory starts by the MCU being loaded with the logical address j to be accessed. The MCU, as shown in Fig. 6 propagates this address to the r individual CCs through a propagation network consisting of cells with a fan-out of two. An alternate network will also be considered for propagation of control. This transmission takes \( \log r \) time units.

The contents of address j are contained in SSE module \( j \mod r \). The SSE module, through a sequence of shuffles and shuffle exchanges,
routes the contents of logical address \( j \) to the access window. This sequence takes \( \log N/r \) time to complete. SSE module \( (j+1) \mod r \) will begin to access logical address \( j+1 \) one time unit after SSE module \( j \mod r \) begins to access \( j \). This behaviour is repeated for SSE module \( j+p \mod r \) with respect to SSE module \( j+p-1 \mod r \) for \( 0 < p < r \). Hence the contents of logical addresses \( j, j+1, \ldots, j+r-1 \) are routed into the access windows of the modules in which they reside, in time sequence. Since addresses are interleaved, these addresses will necessarily lie in different memory modules.

A slight complication occurs if sequential access begins at SSE module \( i > 0 \), since within all memory modules \( k < i \), address \( |j/r| + 1 \) is to be accessed. In order to do this the CC of the \( k \)th SSE module adds 1 to \( |j/r| \) if \( k < j \mod r \). As described in section II this address would be stored in the A-register of the CC.

The contents of the \( r \) logical addresses thus accessed are parallel loaded into the buffer registers \( D_1, D_2, \ldots, D_r \), as in Fig. 6. This parallel loading can be done simultaneously with the last transfer in the shuffle/shuffle exchange access sequence. Contents of the \( j+k \)th address will be stored in buffer \( D_{j+k \mod r} \) for \( 0 \leq k < r \). The data from these buffer registers are to be delivered to the outside world through a single i/o port at the rate of one per time unit. SSE module \( j \mod r \) must begin routing the contents of logical address \( j+r \) to the access window \( r \) time units after the initiation of the access sequence to access contents of logical address \( j \). Since \( r \geq \log N/r \) this access sequence would have been completed before the next access sequence is
initiated. The A-register in each CC is incremented by 1, prior to the initiation of the next access. This behaviour is repeated for the remaining SSE modules. For ease of implementation of the control mechanism N/r is chosen be a power of 2. This restricts r to a power of 2.

Still unresolved now is the problem of routing the data from the r buffer registers to the outside world and leaving the buffer registers free to accept the contents of the next r consecutive logical addresses delivered by the SSE modules. One method, already considered in section III, is to use a multiplexer and demultiplexer to implement random access at this level. In this section it is assumed that r, which has to satisfy the constraint $r \geq \log N - \log r$, is too large to make use of the multiplexer feasible. A second method is to use a binary interconnection network as indicated in Fig. 6. In comparison to the scheme to be proposed the binary interconnection uses more extra cells. It is not clear what effect this would have on the space required to layout the whole memory on a chip.

The problem can be solved by the deployment of a multi-level deck interconnection. In this scheme (let $r = r_1$) the buffer registers $D^1_{1}, \ldots, D^1_{r_1}$ are themselves interconnected by a deck, hence forming a second-level deck, as in Fig. 7. Buffer registers $D^2_{1}$ at the output of the second-level deck may again be connected together to form a third-level deck. Passage of control information to the first-level deck still takes $\log r_1$ time.
Define

\[ f(n) = \inf_p \{ p \geq \log n - \log p \text{ and } p \text{ is a power of } 2 \}. \]

It can be shown by induction that for all \( n, k > 0 \)

\[ f(n) = 2^k \iff 2^{k-1+k} \leq n \leq 2^{k+k}. \]

Hence \( f(n) = 2^k \Rightarrow n^2 > 2^{2k}; \)

\[ \text{i.e.,} \]

\[ n^2 > 2^{2f(n)}. \]

Thus \( f(n) < \log n^2 = 2 \log n. \)

These \( r_1 \) cells are divided into \( r_2 \) sets to form the second-level deck.

Minimisation of \( r_1 \) is required primarily to reduce random access time.

Furthermore to reduce the amount of interconnection in the higher level decks \( r_1 \) has to be minimised for \( i > 0 \). Since the \( i \) th-level deck is divided into \( r_1 \) sets of registers, it can be shown that \( r_1 \) has to be restricted to a power of 2, thus simplifying control at all levels of the deck.

A sufficient condition to ensure unit sequential time is for \( r_1 \)
to satisfy the inequality \( r_1 > \log r_i - \log r_i \) for \( i > 0 \), with \( r_0 = N \).

This requirement is met by setting \( r_1 = f(r_{i-1}) \). If \( f_i(N) \) represents the result of the recursive application of the functional \( f(.) \) \( i \) times on \( N \) \( (f^0(N) = N) \), then the random access time for this multi-level deck interconnection is \( \log f(N) + \sum_{i=0}^{k} \log f_i^{-1}(N) - \log f_i(N) = \log f(N) + \log N - \log f_k(N) \), where the multi-level deck consists of \( k \) levels. \( f_k(N) \)
represents the number of buffers at the highest level.
k is chosen so that \( f^k(N) \) is small enough to justify the use of \( f^k(N) \)-to-1 multiplexer and 1-to-\( f^k(N) \) demultiplexer. Alternately \( f^k(N) \) can be made to go to 1, in which case \( \log f^k(N) = 0 \). This shall be assumed to be true in the deck under consideration. Hence random access time is bounded from above by \( \log N + \log f(N) \leq \log N + \log \log(N) + 1 \), so that the random access time is at most \( \log N + \log \log N \). It should be noted that no additional time is spent in transferring control information to the \( i \)th-level deck for \( i > 1 \), since this control information can be transferred from the \( i-1 \)th-level deck control circuitry while data is still being shuffled in the \( i-1 \)th-level deck.

The \( i \)th-level deck delivers data into its buffers at the rate of one every time unit. After delivering a data item the SSE module at the \( i \)th-level deck is permitted to idle for \( f^i(N) - (\log f^{i-1}(N) - \log f^i(N)) \) time units. During this period the SSE module can execute any sequence of shuffles and shuffle exchanges. This ensures that data is output at the average rate of one per time unit after the initial delay for each level of deck. A simple inductive argument can be constructed to show that the network meets the desired unit sequential access rate beyond the initial delay for random access.

The time for a random write access is less than the time for a random read access. The control information would still take \( \log f(N) \) time to propagate to the control units. By overlapping the execution of the access sequence in the SSE modules of the lowest level deck with the transmission of data from the i/o port to the lowest level deck the random access time can be shown to be \( \log N \). Subsequent sequential writes take unit time. Analysis of the performance of the memory under
heavily loaded conditions has been carried out. Some of the activity of successive accesses can be overlapped. Under assumptions that successive access requests are independent and that the access requests consist of both reads and writes it can be shown that the average delay between successive accesses is at bounded from above by \((\log N - \log f(N))/2 + 17/32\). Further decrease of this delay is possible.

Use has been made of the binary propagation network to propagate the control information. A multi-level deck with broadcast capability at each cell could have served the purpose adequately.

The average fan-out from a memory cell does not increase significantly over 2. An additional \(O(\log N)\) memory cells are used in the multilevel deck.

VI CONCLUSION

This paper has shown how worst case random access property of dynamic memories can be improved at the cost of additional control circuitry. For a dynamic memory of \(N\) elements partitioned into \(r\) sets a memory interconnection network has been proposed that permits random access of any element in \(\log (N/r)\) time units and subsequent sequential access in unit time. This memory organisation has been shown feasible for even large memories. Asymptotic behaviour has been handled by modifying the proposed interconnection.
REFERENCES.


N = 8
Shuffle Exchange Interconnection

Fig. 2.

N = 8
Shuffle Interconnection

Fig. 1.
A, B, & C denote the contents of registers A, B, & C respectively.

* denotes logical left shift.

+ denotes the exclusive or operation.

Fig. 3
$N = 32$
$r = 4$

Deck Interconnection

Fig. 4

Control Circuitry for the Deck

Fig. 5
Binary Interconnection Network as an Output Interface for the Deck

Fig. 6
Multi Level Deck

Fig. 7
<table>
<thead>
<tr>
<th>r</th>
<th>log r</th>
<th>log N</th>
<th>N</th>
<th>RAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$2^1$</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>$2^3$</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>6</td>
<td>$2^6$</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>11</td>
<td>$2^{11}$</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>20</td>
<td>$2^{20}$</td>
<td>16</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
<td>37</td>
<td>$2^{37}$</td>
<td>32</td>
</tr>
</tbody>
</table>

RAT : Random Access Time

Random Access Times in Decks

Table 1.