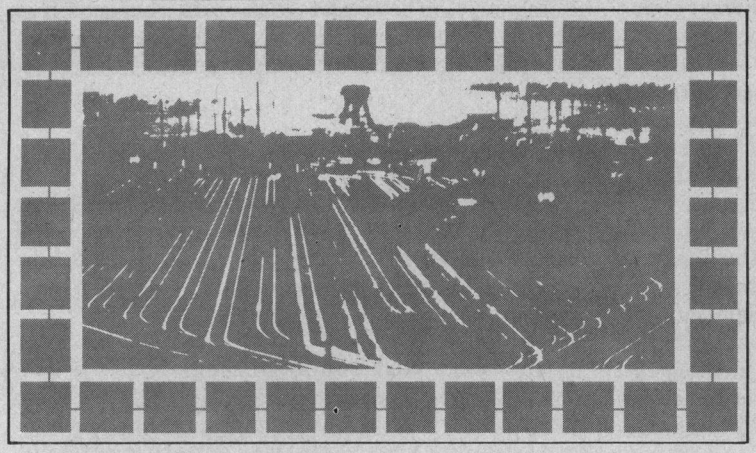
Many SIMD interconnection networks have been proposed. To put the different approaches into perspective, this analysis compares a number of single- and multistage networks.



Interconnection Networks for SIMD Machines

Howard Jay Siegel
Purdue University

With the advent of microprocessors, large-scale parallel processing systems with as many as 2^{14} to 2^{16} processors have become feasible.[1,2] One multimicroprocessor structure—the SIMD (single instruction stream-multiple data stream) mode of parallel processing—is especially suitable for exploiting the parallelism inherent in certain tasks. SIMD systems are currently being used for scientific operations such as matrix calculations[3] and image processing.[4,5] In the future, their scope may be broadened to include business calculations that require the same program to be executed on many different data sets—e.g., computing bank-account interest.

Two existing SIMD machines are the Illiac IV[6] and Staran,[7] and many new ones have been proposed.[8-16] In designing SIMD systems, constructing an interconnection network for communications among the processors and memories presents a major problem. In this article, we analyze different approaches to network design, limiting our discussion to SIMD machine networks (other types are explored elsewhere[17-19]).

Interconnecting N processors and N memory modules in an SIMD system where N may be 2^6 to 2^{16} is a non-trivial task. A single shared bus is not sufficient, since in an SIMD machine it is desirable to allow many processors to send data to other processors simultaneously (e.g., from processor i to processor $i+1$, $0 \leq i < N-1$). Ideally, one would like each processor directly to every other processor, but this is highly impractical for large N, since each processor would require $N-1$ lines. An alternate network which allows all processors to communicate simultaneously is the crossbar switch.[20] The difficulty here is that network costs grow with N^2; given current technology, this makes crossbar switches infeasible for large systems.

To solve the problem of providing fast, efficient communiations at a reasonable cost, many different networks have been proposed in the literature, a number of which we discuss here. However, no single network is generally considered "best," since the cost-effectiveness of a particular design varies with such factors as the computational tasks for which it will be used, the desired speed of interprocessor data transfers, the actual hardware implementation of the network, the number of processors in the system, and the cost constraints on the construction.

SIMD machines

The acronym *SIMD* stands for *single instruction stream-multiple data stream.*[21] Typically, an *SIMD machine*[22-25] is a computer system consisting of a control unit, N processors, N memory modules, and an interconnection network. The control unit broadcasts instructions to all processors, and all active processors execute the same instruction at the same time. Thus, there is a single instruction stream. Each active processor executes the instruction on data in its own associated memory module. Thus, there is a multiple data stream. The interconnection network, sometimes referred to as an alignment or permutation network, provides a communications facility for the processors and memory modules.

Structure. The physical structure of an SIMD machine can be viewed as a set of N processing elements (PEs), where each PE consists of a processor with its own memory (e.g., Illiac IV[6]). The network connects each PE to some subset of the other PEs. A transfer instruction causes data to be moved from each PE to one of the PEs to which it is con-

0018-9162/79/0600-0057$00.75

nected. To move data between two PEs that are not directly connected, the data must be passed through intermediary PEs by executing a programmed sequence of data transfers.

An alternate structure is to position the network between the processors and the memories (e.g., Staran[7]). In general, in the processor-to-memory configuration the network connects each processor to some subset of memories. A transfer instruction causes data to be moved from each processor to one of the memories to which it is connected, or from each memory to one of the processors to which it is connected. One processor can transfer data to another through any memory connected to both. To pass data between two processors, a programmed sequence of data transfers that moves the data through intermediary memories and processors must be executed. Variations on the processor-to-memory configuration include (1) using two networks, one for processor-to-memory and the other for memory-to-processor communications,[26] and (2) using the same network for processor-to-memory and processor-to-processor communications.[27]

Operation. To demonstrate how SIMD machines operate, let us consider two examples—matrix operations and image processing—using a PE-to-PE configuration of N PEs.

Assume that three N-word vectors, A, B, and C, are stored such that PE i contains $A(i)$, $B(i)$, and $C(i)$, $0 \leqslant i < N$. To add the elements of vectors A and B and store the result in C, all PEs would execute $C=A+B$, with PE i doing the addition for elements $A(i)$ and $B(i)$. Thus, the SIMD machine does in one step a task requiring N steps on a serial processor.

As an example of why the interconnection network is needed, consider the N-step serial task:

for $i=1$ until $N-1$ do $C(i) = A(i) + B(i-1)$
$C(0) = A(0)$

An SIMD machine does this task in three steps: (1) it moves the value of $B(i-1)$ to PE i from PE $i-1$, $1 \leqslant i < N$, via the interconnection network; (2) in PE i, it adds $A(i)$ to $B(i-1)$ and stores the result in $C(i)$, $1 \leqslant i < N$; and (3) in PE 0, it stores $A(0)$ in $C(0)$. (At each step certain PEs will be disabled.) Assuming the appropriate interconnections are available, PE $i-1$ sends its B data to PE i for all i, $1 \leqslant i < N$, simultaneously. Thus, Step 1 requires only one parallel data transfer. (Information about the use of SIMD machines for more complex matrix calculations can be found in Stone.[3])

As a simple example of image processing, consider smoothing, i.e., replacing point (i,j) of an image with the average value of (i,j) and its eight surrounding points: $(i-1,j)$, $(i-1,j-1)$, $(i,j-1)$, $(i+1,j)$, $(i+1,j+1)$, $(i,j+1)$, $(i-1,j+1)$, and $(i+1,j-1)$. Given a 2^k by 2^k image and a serial processor, $2^k * 2^k = 2^{2k}$ averages must be computed. Given an SIMD machine with $N = 2^n$ processors, the image can be partitioned into a $\sqrt{N}$ by $\sqrt{N}$ array of subpictures of size $2^k/\sqrt{N}$ by $2^k/\sqrt{N}$. All N subpictures can be smoothed in parallel, requiring the calculation of $2^{2k}/N$ averages, where for each of these calculations the new values for N points are computed in parallel. In general, however, calculating the averages for the edge points in a subpicture requires obtaining data from "adjacent" subpictures stored in other PEs. This necessitates the use of the interconnection network to move data from one PE to another. (Siegel[28] mentions other image processing tasks possible with SIMD machines.)

Multistage SIMD network parameters

One type of interconnection network for SIMD machines consists of many stages of interconnected switches. Many *multistage SIMD networks* can be described by three parameters: interchange box, topology, and control structure.

Interchange box. The interchange box is a two-input two-output device used as a basic building block in many multistage networks. Let the upper input and output lines be labeled i and the lower input and output lines be labeled j. The four legitimate states of an interchange box are (1) straight—input i to output i, input j to output j; (2) exchange—input i to output j, input j to output i; (3) lower broadcast—input j to outputs i and j; and (4) upper broadcast—input i to outputs i and j.[26] A *two-function interchange box* is capable of being in either the straight or exchange states, and a *four-function interchange box* is capable of being in any of the four legitimate states.[29]

Topology. Network topology is the actual interconnection pattern used to connect a set of N input lines to N output lines. Generally, multistage networks consist of n stages, where $N=2^n$ is the number of input and output lines. Conceptually, each stage consists of an interconnection pattern of N lines and $N/2$ interchange boxes.

One type of topology is called the cube. In the three-dimensional cube shown in Figure 1, horizontal lines connect vertices whose labels differ in the zeroth (low-order) bit position, diagonal lines connect vertices which differ in the first (middle) bit position, and vertical lines connect vertices which differ in the second (high-order) bit position. This can be generalized for an n-dimensional cube.

An example of a cube-type multistage network topology is the Staran network.[27] This network can be drawn using interchange boxes, as shown in Figure 2. For each interchange box, the upper and lower outputs are labeled with the same numbers as the upper and lower inputs, respectively. At stage i, the two input lines that differ only in their ith bit position are paired together as inputs to an interchange box, for $0 \leqslant i < n$. The data first pass through stage 0, then 1, etc. At each stage, the interchange boxes connect their input lines to their output lines to complete the connections from one end of the network to the other.

Control structure. The control structure of a network determines how the states of the interchange

boxes will be set.[29] *Individual stage control* uses the same control signal to set the state of all the interchange boxes in a stage (i.e., all the boxes in a given stage must be in the same state). Each stage has its own signal. *Individual box control* uses a separate control signal to set the state of each interchange box. *Partial stage control* uses $i+1$ control signals to control stage i, $0 \leq i < n$. In Staran, this is used for "shift permutations," as will be discussed later.

Multistage SIMD network comparisons

The three classifying parameters defined above will be used to analyze four multistage networks—Staran, indirect binary n-cube, omega, and data manipulator—that have appeared in the literature.

Staran. Conceptually, the Staran network[27] consists of n stages of $N/2$ two-function interchange boxes. Input lines which differ in the ith bit position are paired in the ith stage of the network (Figure 2). This network is used in the Staran SIMD machine manufactured by Goodyear Aerospace,[7] where each array module consists of 256 processors and memories ($N=256$). It is used in two ways: (1) to let processors communicate with other processors and (2) to let processors access memories. The network has two control mechanisms, the flip control and the shift control.

Under the flip (individual stage) control, an n-bit vector $F = f_{n-1} \cdots f_1 f_0$ determines the way stages will be set. If $f_i = 1$, stage i interchange boxes are in the exchange state, and if $f_i = 0$, they are in the straight state. For example, if $N=8$, and $F=010$, network input line $i_2 i_1 i_0$ is connected to network output line $i_2 \bar{i}_1 i_0$. The shift (partial stage) control allows shifts of data from input x to output $x + 2^m$ modulo 2^P, $0 \leq m \leq P \leq n$, using $i+1$ control lines at stage i, $0 \leq i < n$. For $N=8$, line $0A$ controls boxes a, b, c, and d, $1A$ controls e and g, $1B$ controls f and h, $2A$ controls i, $2B$ controls j, and $2C$ controls k and l (Figure 2). For example, to shift data from input x to output $x+1$ modulo 8, set $0A = 1A = 2A =$ exchange and $1B = 2B = 2C =$ straight. The flip control can be implemented by controlling all the shift control lines for a given stage by a single signal. The network connection capabilities are discussed by Batcher[7,27,30] and Bauer.[31]

Indirect binary n-cube. Another cube-type network that has been proposed is the indirect binary n-cube[1] network consisting of n stages of $N/2$ two-function interchange boxes under independent box control. The addresses of the two input lines to an interchange box at stage i differ only in the ith bit position, as shown in Figure 2. The stages are ordered so that data pass through stage 0, then stage 1, etc., and finally through stage $n-1$. The various connection capabilities of this network are discussed by Pease.[1]

The Staran network and the n-cube network have the same topology, and both use two-function interchange boxes. The differences in their capabilities result from their different control schemes. The individual box control of the n-cube is much more flexible than Staran's partial stage control. The n-cube can perform all of the connections that Staran can and some it cannot (e.g., connect input 0 to output 1 and input 2 to output 0). Pease gives us a theorem which may be used to determine if a particular arrangement of I/O connections can be made.[1] Not all connections are possible (e.g., input 0 to output 2 and input 1 to output 4).

Omega. Lawrie's omega network is a multistage SIMD machine network based on the perfect-shuffle interconnection pattern.[26] The perfect-shuffle connection pattern routes data from position P, whose binary representation is $p_{n-1} \cdots p_1 p_0$, to position P', whose binary representation is $p_{n-2} \cdots p_1 p_0 p_{n-1}$. Let $S(P)$ represent the "shuffle" of P. Then, the data in position P is routed to position $S(P)$, where

$$S(p_{n-1} \cdots p_1 p_0) = p_{n-2} \cdots p_1 p_0 p_{n-1}.$$

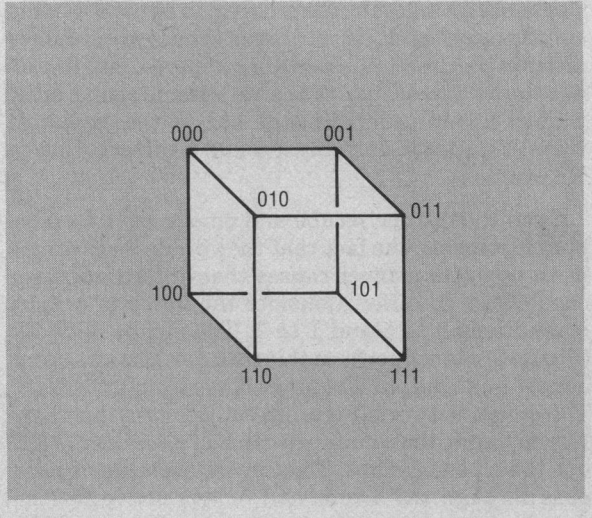


Figure 1. Three dimensional cube structure, with vertices labeled from 0 to 7 in binary.

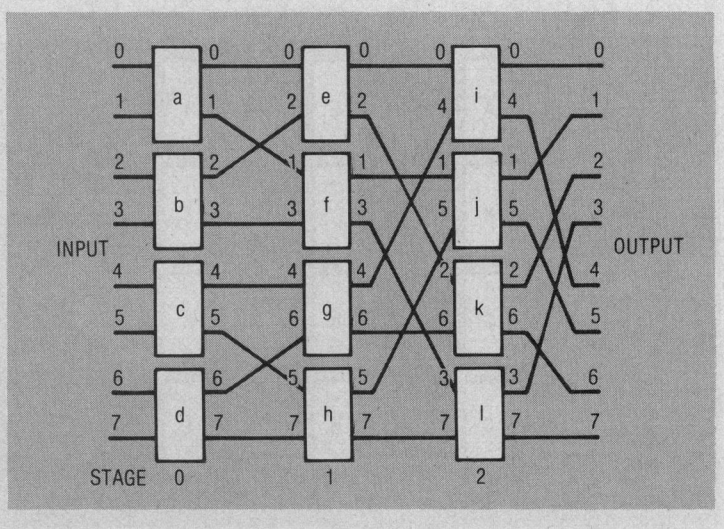


Figure 2. Staran network based on interchange boxes, for $N = 8$. (Also, indirect binary n-cube topology.)

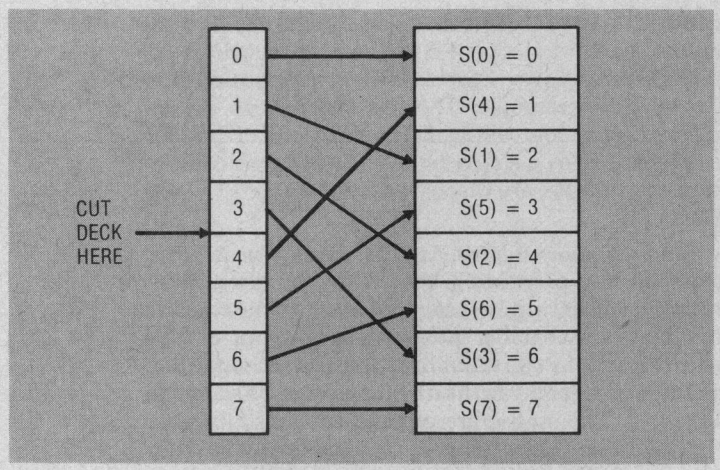


Figure 3. Perfectly shuffling a deck of eight cards.

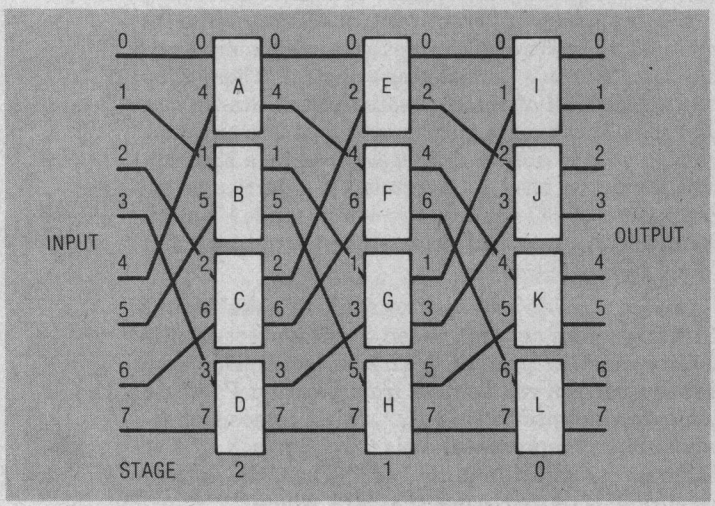


Figure 4. Omega network for $N = 8$.

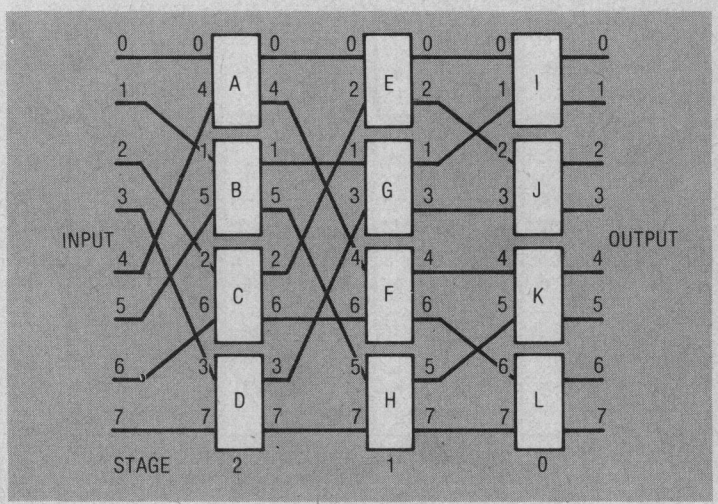


Figure 5. Omega network for $N = 8$ redrawn.

The name "perfect shuffle" has its origin in shuffling cards—i.e., perfectly intermixing two halves of a deck as shown in Figure 3. The mathematical properties of the shuffle are discussed by Golomb[32] and Johnson.[33] Its use in parallel processing has been examined by Stone.[34]

An N by N omega network consists of n identical stages, where each stage is a perfect-shuffle interconnection followed by a column of $N/2$ four-function interchange boxes under independent box control (Figure 4). The various interconnections performed by the omega network are examined by Lawrie.[26] A multistage "shuffle-exchange" network, similar to the omega network, and its capabilities are presented by Lang and Stone.[35]

The interchange boxes of the omega network can be repositioned, without modifying the interchange box interconnections, as shown in Figure 5. Thus, the networks of Figures 4 and 5 are equivalent. The Staran/n-cube topology of Figure 2 is identical to the omega topology of Figure 5 with the exception of the order of stages through which data pass. In the Staran/n-cube network, data pass through stage 0, then stage 1, etc., while in the omega network data pass through stage $n-1$, then stage $n-2$, etc. Thus, the two differences between the omega network and the n-cube network are the order in which the stages are traversed and the type of interchange box—i.e., four-function versus two-function.

The four-function interchange boxes of the omega network allow "one-to-many" connections. For example, network input 2 can be broadcast to all network outputs in one pass through the network, by setting boxes E and F to lower broadcast and C, I, J, K, and L to upper broadcast. The Staran and n-cube networks are limited to one-to-one connections since they use two-function interchange boxes.

To examine the effect of the reversed stage order of the n-cube and omega networks, consider only the exchange and straight states of the four-function interchange boxes. Since the topologies of these two networks are isomorphic, the omega can be considered a cube-type network, i.e., at stage i, inputs whose labels differ in the ith bit position are compared, and to connect input I to output O, the data from input I must pass through an interchange box in the exchange state at stage i if, and only if, I and O differ in the ith bit position.

Even though the n-cube and omega have isomorphic topologies, the fact that they order their stages in an opposite manner causes their interconnection capabilities to differ. Consider the input to output connections 0 to 5 and 1 to 7. The n-cube network (Figure 2) cannot perform this task (both inputs 0 and 1 must go to output 1 at stage 0, creating a conflict). The omega network (Figure 4) can perform this task. Furthermore, the n-cube can connect 5 to 0 and 7 to 1, but the omega cannot. This occurs because, as data pass through the n-cube and omega networks, the data items that are paired in an interchange box depend on the order of the stages of the network, i.e.,

stage i being encountered as the ith stage or as the $n-(i+1)$ stage, $0 \leq i < n$.

Consider an arbitrary set of input and output connections which the n-cube can perform. To perform these connections on the omega, it is necessary to transform the number (address) of each input and output line from $p_{n-1} \cdots p_1 p_0$ to $p_0 p_1 \cdots p_{n-1}$.[29] This implies, for example, that if the omega network is used to interconnect N PEs and one wishes to execute an SIMD machine program written for a system with the n-cube network, then PE $P = p_{n-1} \cdots p_1 p_0$ must act as if it were PE $p_0 p_1 \cdots p_{n-1}$, $0 \leq P < N$.

To perform the 5 to 0 and 7 to 1 connections that the n-cube can make but the omega cannot, for $N=8$, PE 4 must act as PE 1 (under the transformation, PE addresses 0, 5, and 7 are unchanged). The omega network can connect 5 to 0 and 7 to 4. Similarly, the n-cube can perform the omega connections 0 to 5 and 1 to 7 if PE 4 acts as PE 1. The correctness of this address transformation method can be proved formally by realizing that the renumbering causes the omega network first to pair data lines whose numbers differ only in the zeroth bit position, then the first bit position, etc., just the way the n-cube does.

Thus, considering only one-to-one connections, by using the relabeling technique the n-cube and omega networks are functionally equivalent. That is, by relabeling the processors and memories and loading the data into the memories based on their new labels, an SIMD progam written to run on one network can be run on the other. Since the omega network is functionally equivalent to the n-cube, it can (with relabeling) perform all the interconnections of the Staran network. Furthermore, because the omega network has four-function interchange boxes, it can perform one-to-many connections which the n-cube cannot.

Augmented data manipulator. The data manipulator network, introduced by Feng,[36] consists of n stages connecting columns of N cells, as shown in Figure 6. For $0 \leq j < N$ and $0 \leq i < n$, there are three sets of interconnections at stage i: one sends the data from input cell j to output cell $j + 2^i$ modulo N, one sends the data from input cell j to output cell $j - 2^i$ modulo N, and one sends the data from input cell j to output cell j. Stage i of the network is controlled by a pair of signals, two of H_1^i, H_2^i, U_1^i, U_2^i, D_1^i, and D_2^i, as specified in Figure 6. The topology is such that at stage i input cell j is connected to output cell j and to the output cell which differs from j in only the ith-bit position, just as the topologies of the omega and n-cube allow. In addition, the data manipulator connects j to both $j + 2^i$ modulo N and $j - 2^i$ modulo N, one of which differs from j in more than the ith bit position. The stages are ordered $n-1, n-2, \ldots 0$, as in the omega network topology.

With control structure modifications, the data manipulator is capable of all the interconnections that the omega network achieves with four-function boxes and individual box control, plus some additional ones. The augmented data manipulator, or ADM, is a data manipulator with individual cell control, i.e., each cell will get its own control signals.[29] Specifically, each cell can get any of the signals H, U, and D.

Consider simulating the omega four-function box at stage i with upper input x and lower input y. Let the subscript x denote the ADM control for input cell x at stage i, and let the subscript y be used similarly. Then, to simulate the four possible states of an interchange box use the following control signals: straight, $H_x H_y$; exchange, $D_x U_y$; lower broadcast, $U_y H_y$; and upper broadcast, $D_x H_x$. Thus, the ADM can perform any interconnection that the omega network can. However, the omega network can perform a maximum of $2^{Nn/2}$ one-to-one connections, i.e., each box can be straight or exchange. The ADM can do all of these connections and in addition can, for example, connect 0 to 7, 7 to 0, and i to i, for $0 \leq i < N$, $i \neq 0,7$, when N equals 8. Thus, the number of one-to-one connections the ADM can perform exceeds the maximum physical limit for the omega. Other properties and features of the data manipulator and ADM networks are discussed by Feng[36] and Siegel and Smith.[29,37]

Network flexibility. Thus, in terms of interconnection capabilities, the four multistage networks which have been discussed can be ordered in terms of increasing flexibility as follows: Staran, indirect binary n-cube, omega, and ADM. It should be noted, however, that increased flexibility incurs increased cost, and the Staran is the only one of the four networks that has been constructed and used for a large value of N. The most cost-effective network for a particular application will depend on a variety of factors

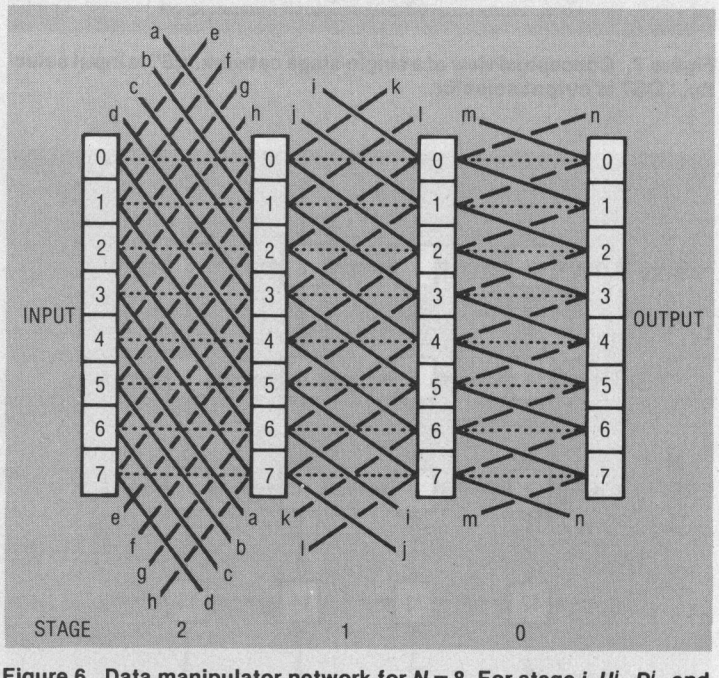


Figure 6. Data manipulator network for $N = 8$. For stage i, U_1^i, D_1^i, and H_1^i control those cells whose ith bit is 0, and U_2^i, D_2^i, and H_2^i control those cells whose ith bit is 1. U means use dashed line connection, D means use solid line connection, and H means use dotted line connection.

mentioned earlier, such as tasks to be performed and data transfer speed requirements.

The cube-type multistage network topology has also been called an SW banyan.[38] Further information about the relationships among different multistage interconnection networks can be found in Siegel and Smith[29,39] and Wu and Feng.[40]

Single-stage SIMD networks

As an alternative to multistage networks, with n or more stages of switches, a single stage of switches may be used, forming a single-stage SIMD network.

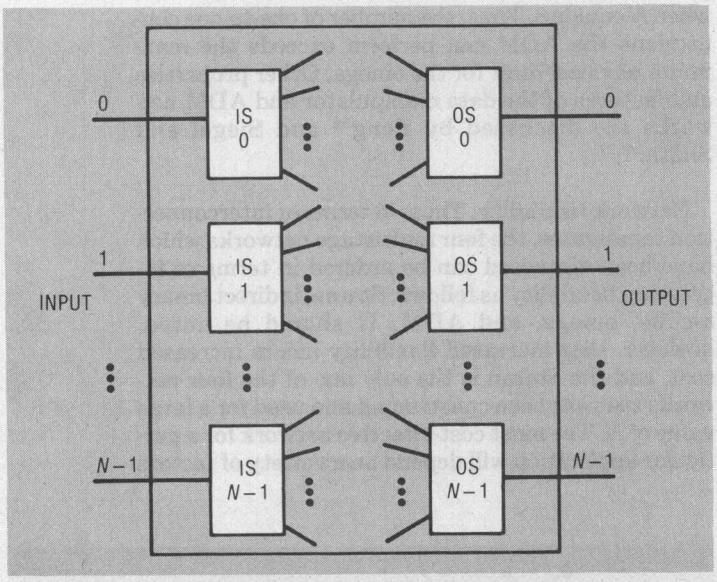


Figure 7. Conceptual view of a single-stage network. "IS" is input selector, "OS" is output selector.

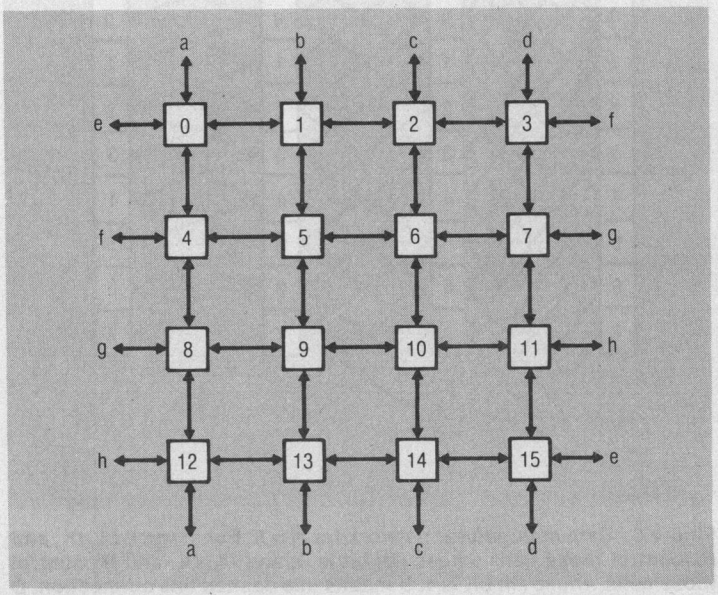


Figure 8. Illiac network for $N = 16$ (Illiac IV has $N = 64$). Vertical lines are $+\sqrt{N}$ and $-\sqrt{N}$. Horizontal lines are $+1$ and -1.

Conceptually, a *single-stage network* may be viewed as N input selectors and N output selectors, as shown in Figure 7. They way in which the input selectors are connected to the output selectors determines the allowable interconnections. Four single-stage networks will be discussed here: the Illiac, PM2I, shuffle-exchange, and cube.

Illiac. The Illiac network, used in the Illiac IV SIMD machine,[6] allows PE i to send data to any one of PE $i+1$, PE $i-1$, PE $i+\sqrt{N}$, or PE $i-\sqrt{N}$, arithmetic modulo N, as shown in Figure 8. This is often referred to as the nearest neighbor or north, south, east, and west connection pattern. Relating this to Figure 7, input selector i has lines to output selectors $i+1, i-1, i+\sqrt{N}$, and $i-\sqrt{N}$. Output selector j gets its inputs from $j-1$, $j+1$, $j-\sqrt{N}$, and $j+\sqrt{N}$. Since the Illiac is a single-instruction-stream machine, all active PEs must use the same connection at the same time. Various properties and abilities of this network are discussed in the literature.[6,41-44]

PM2I. The Plus-Minus 2^i network[42] is a generalization of the Illiac network which allows PE j to send data to any one of PE $j+2^i$ or PE $j-2^i$, $0 \leq i < n$, arithmetic modulo N, as shown in Figure 9. The PM2I connection patterns form the basis for the data manipulator[36] and the ADM[29] multistage networks, i.e., stage i uses the $j+2^i$ and $j-2^i$ connections. In terms of I/O selectors, input selector j is connected to output selectors $j+2^i$ and $j-2^i$, $0 \leq i < n$. Output selector j gets its inputs from input selectors $j-2^i$ and $j+2^i$, $0 \leq i < n$. As with the Illiac network, all active PEs must use the same PM2I interconnection at the same time. For example, if one PE is using the $+2^0$ connection, all active PEs must use their $+2^0$ connection. The connection capabilities and other properties of the PM2I network are discussed in Siegel.[42-44]

Shuffle-exchange. The shuffle-exchange network[3,34] is based on the perfect-shuffle connection (Figure 3) defined earlier. When the perfect shuffle is used as a connection in a single-stage network, it is accompanied by an exchange connection. This is due to certain limitations of the shuffle connection, e.g., PE 0 cannot communicate with any other PE. The exchange interconnection allows each PE to send data to the PE whose address differs from it in only the low-order bit position. The shuffle-exchange connections are the basis for the omega network,[26] where the shuffle connection precedes each stage of switches and the action of the interchange boxes can implement the exchange. For the shuffle-exchange single-stage network shown in Figure 10, input selector $p_{n-1} \cdots p_1 p_0$ is connected to output selectors $p_{n-2} \cdots p_1 p_0 p_{n-1}$ and $p_{n-1} \cdots p_1 \bar{p}_0$. Output selector $g_{n-1} \cdots g_1 g_0$ gets its inputs from input selectors $g_0 g_{n-1} \cdots g_2 g_1$ and $g_{n-1} \cdots g_1 \bar{g}_0$. As with the other networks, all active PEs must use the same type of connection at the same time. For example, if one PE "shuffles," all active PEs must shuffle. The usefulness and attributes of this network are examined in Lang,[45] Siegel,[42-44] and Stone.[3,34]

Cube. The cube network is a generalization of the exchange connection, allowing each PE to communicate with any of the n PEs whose address differs from it in any one bit position.[42] Thus, PE P, whose binary representation is $p_{n-1}\cdots p_1p_0$, can send data directly to any one of PE $p_{n-1}\cdots p_{i+1}\bar{p}_ip_{i-1}\cdots p_0$, $0\leq i<n$ (Figure 1). These cube connections are the basis of the Staran[27] and n-cube[1] multistage networks, where stage i allows PE $p_{n-1}\cdots p_1p_0$ to send data to PE $p_{n-1}\cdots p_{i+1}\bar{p}_ip_{i-1}\cdots p_0$, $0\leq i<n$. For the cube single-stage network, shown in Figure 11, input selector $p_{n-1}\cdots p_1p_0$ is connected to output selectors $p_{n-1}\cdots p_{i+1}\bar{p}_ip_{i-1}\cdots p_0$, $0\leq i<n$. Output selector $g_{n-1}\cdots g_1g_0$ gets its inputs from input selectors $g_{n-1}\cdots g_{i+1}\bar{g}_ig_{i-1}\cdots g_0$, $0\leq i<n$. As with the other networks, only one type of cube connection, (i.e., a fixed value of i) can be used at a time. The features and abilities of the cube network are discussed in Siegel.[42-44]

Single-stage vs. multistage networks. Single-stage networks are also called recirculating networks because data items may have to recirculate through the single stage several times before reaching their final destinations.[37] To expand on this, consider using a single-stage network to simulate the multistage network to which it is related.

The single-stage shuffle-exchange network can be used to simulate the omega multistage network. For each stage of the omega network configuration being simulated, have all PEs use the shuffle interconnection. Then have PE j use the exchange interconnection if at that stage line j is the input to an interchange box in the exchange state or is the input being broadcast in one of the broadcast states, $0\leq j<N$. (To prevent a PE from using an interconnection, it must be turned off (disabled) for the execution of that transfer instruction.) Thus, the single-stage shuffle-exchange can simulate one pass through the omega network in at most $2n$ recirculations.

The n-cube and Staran multistage networks can be simulated by the single-stage cube network in at most n recirculations per pass. For $i=0,1,\ldots,n-1$, if input line j goes to an interchange box in the exchange state at stage i, PE j should use the cube interconnection that sends its data to the PE that differs from j in the ith bit position. Note that if, for example, all of the interchange boxes in the n-cube network are set to straight except for those in a single stage, then only one pass through the cube single-stage network would be required.

The PM2I network can simulate one pass through the ADM multistage network in at most $2n$ recirculations. For $i=n-1,n-2,\ldots,0$, have PE j use the $+2^i$ PM2I interconnection if at stage i cell j in the ADM network configuration being simulated gets the D control signal. Then use the -2^i PM2I interconnection if cell j gets the U control signal. The number of recirculations needed to simulate a stage of a particular ADM configuration depends on the control signals going to its cells: (1) if there are no U or D signals, nothing needs to be transferred in the single-stage simulation; (2) if there are both Us and Ds, then two recirculations are needed; and (3) otherwise, only one pass is needed.

Single-stage network simulations. To compare different single-stage networks with each other, upper and lower bounds on the number of parallel interprocessor data transfers needed for each network to simulate the others have been studied.[42,43] The bounds are based on the interconnection (of the network being simulated) which requires the most

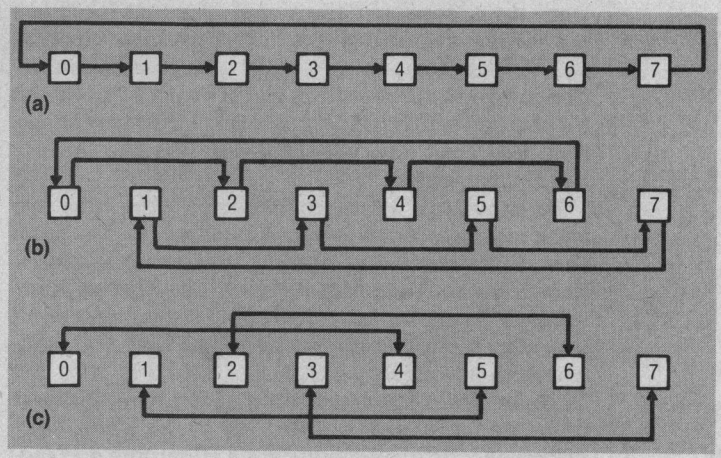


Figure 9. PM2I single stage network for $N=8$: (a) $+2^0$ connection; (b) $+2^1$ connection; and (c) $+2^2$ connection. For the -2^i connections, reverse the directions of the arrows.

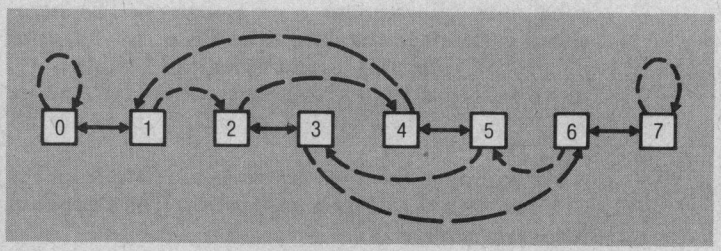


Figure 10. Shuffle-exchange single-stage network for $N=8$. Solid line is exchange, dashed line is shuffle.

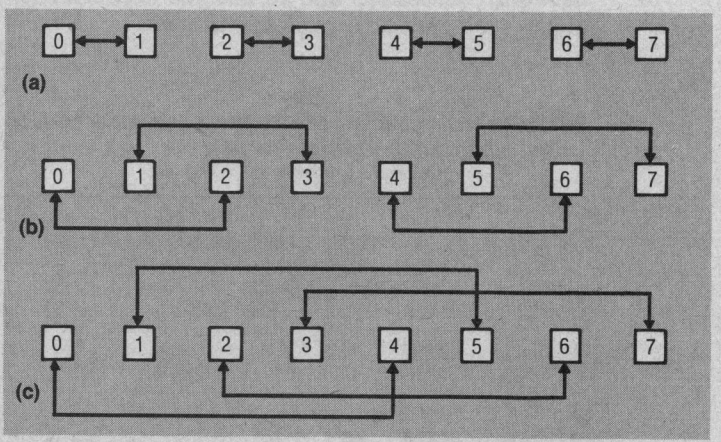


Figure 11. Cube single-stage network for $N=8$: (a) transfer based on low-order bit, (b) transfer based on middle bit, and (c) transfer based on high-order bit.

transfers to simulate. Each upper bound is shown using an algorithm to do the simulation. To demonstrate the techniques used, the simulation of a cube interconnection by the PM2I network and a PM2I interconnection by the cube network are discussed. The "cube by PM2I" simulation is directly related to the "omega by ADM" simulation, and the "PM2I by cube" simulation is directly related to the shift control scheme used in the Staran.

In general, a cube interconnection cannot be simulated by a single PM2I connection (this follows from the networks' definitions). Thus, at least two transfers are required. To simulate the PE $p_{n-1}\cdots p_1p_0$ to $p_{n-1}\cdots p_{i+1}\bar{p}_ip_{i-1}\cdots p_0$ cube connection in two steps, first have all PEs with a 0 in the ith bit position of their address use the $+2^i$ PM2I connection, and then have all the other PEs use the -2^i PM2I connection.

To simulate a PM2I connection by the cube network requires n transfers in the worst case. The proof that n transfers are needed involves the metric known as the Hamming distance. Let d be the Hamming distance, i.e., let $d(x,y)$ equal the number of bit positions in which x and y differ. The PM2I $+2^0$ interconnection connects PE 011...1 to PE 100...0. This means that it moves data a Hamming distance of n, i.e., $d(011\ldots1,100\ldots0) = n$. Each cube interconnection can move data a Hamming distance of only 1, i.e., from a PE to another PE which differs from it in only one bit position. Thus, to move data a Hamming distance of n, n transfers are needed.[42]

To perform any $+2^i$ PM21 interconnection with the cube network requires $n-i$ transfers. The algorithm is (1) move the data from PE $p_{n-1}\cdots p_1p_0$ to $p_{n-1}\cdots p_{i+1}\bar{p}_ip_{i-1}\cdots p_0$ and (2) for $j=i+1$ until $n-1$, move data from PE $p_{n-1}\cdots p_1p_0$ to PE $p_{n-1}\cdots p_{j+1}\bar{p}_j p_{j-1}\cdots p_0$ if bits i to $j-1$ of $p_{n-1}\cdots p_1p_0$ are all zeros.[43]

For example, for $N=8$ and $i=1$, the data from PE 011 goes to 001 and then goes to 101. The algorithm for -2^i is similar.

Table 1 shows the lower and upper bounds for all the simulations.[42,43] The methods for finding these bounds can also be used to study other single-stage network connections and to analyze network suitability for a particular implementation.

Table 1.
Entries in row *i*, column *j* are the lower and upper bounds on the number of transfers for network *i* to simulate network *j*, where $n = \log_2 N$.

		ILLIAC	PM2I	SHUFFLE-EXCHANGE	CUBE
ILLIAC	LOWER	—	$\sqrt{N}/2$	$1+\sqrt{N}/2$	$1+\sqrt{N}/2$
	UPPER	—	$\sqrt{N}/2$	$3\sqrt{N}-4$	$1+\sqrt{N}/2$
PM2I	LOWER	1	—	n	2
	UPPER	1	—	$2n-2$	2
SHUFFLE-EXCHANGE	LOWER	$2n-1$	$2n-1$	—	$n+1$
	UPPER	$2n$	$2n$	—	$n+1$
CUBE	LOWER	n	n	n	—
	UPPER	n	n	n	—

Conclusion

Although many SIMD interconnection networks have been proposed, there is little in the literature examining the relationships between different networks. The comparisons presented here should aid the system designer in choosing an appropriate network for a specific system. As for the future, more and more microprocessor or LSI-based SIMD machines are being proposed. The study, construction, and/or simulation of these different designs will bring new insights into the applications for SIMD machines and the interconnection network features needed to support them. ■

Acknowledgments

The author thanks L. J. Siegel for her comments and suggestions. This work was supported by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under Grant No. AFOSR-78-3581. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

References

1. M. C. Pease, "The Indirect Binary n-Cube Microprocessor Array," *IEEE Trans. Comput.*, Vol. C-26, No. 5, May 1977, pp. 458-473.
2. H. Sullivan, T. R. Bashkow, and K. Klappholz, "A Large Scale Homogeneous, Fully Distributed Parallel Machine," *4th Annual Symp. on Computer Architecture*, Mar. 1977, pp. 105-124.
3. H. S. Stone, "Parallel Computers," in *Introduction to Computer Architecture*, H. S. Stone, editor, Science Research Associates, Chicago, pp. 327-355.
4. A. J. Krygiel, "An Implementation of the Hadamard Transform on the STARAN Associative Array Processors," *1976 Int'l. Conf. on Parallel Processing*, Aug. 1976, p. 34.
5. D. Rohrbacker and J. L. Potter, "Image Processing with the Staran Parallel Computer," *Computer*, Vol. 10, No. 8, Aug. 1977, pp. 54-59.
6. W. J. Bouknight et al., "The Illiac IV System," *Proc. IEEE*, Vol. 60, Apr. 1972, pp. 369-388.
7. K. E. Batcher, "STARAN Parallel Processor System Hardware," *AFIPS Conf. Proc. 1974 NCC*, Vol. 43, May 1974, pp. 405-410.
8. P. M. Flanders et al., "Efficient High Speed Computing with the Distributed Array Processor," *Symp. on High Speed Computer and Algorithm Organization*, Apr. 1977, pp. 113-128.
9. L. Fung, "A Massively Parallel Processing Computer," *Symp. on High Speed Computer and Algorithm Organization*, Apr. 1977, pp. 203-204.
10. J. Keng and K. S. Fu, "A Special Computer Architecture for Image Processing," *1978 IEEE Comp. Soc. Conf. Pattern Recognition and Image Processing*, June 1978, pp. 287-290.
11. G. J. Lipovski, "On a Varistructured Array of Microprocessors," *IEEE Trans. Comp.*, Vol. C-26, No. 2, Feb. 1977, pp. 125-138.

12. G. J. Lipovski and A. Tripathi, "A Reconfigurable Varistructure Array Processor," *1977 Int'l. Conf. on Parallel Processing*, Aug. 1977, pp. 125-138.

13. G. J. Nutt, "Microprocessor Implementation of a Parallel Processor," *4th Annual Symp. on Computer Architecture*, Mar. 1977, pp. 147-152.

14. G. J. Nutt, "A Case Study of Simulation as a Computer System Design Tool," *Computer*, Vol. 11, No. 10, Oct. 1978, pp. 31-36.

15. H. J. Siegel, L. J. Siegel, R. J. McMillen, P. T. Mueller, Jr., and S. D. Smith, *An SIMD/MIMD Multimicroprocessor System for Image Processing and Pattern Recognition*, 1979 IEEE Comp. Soc. Conf. Pattern Recog. and Image Processing, Aug. 1979.

16. H. J. Siegel, P. T. Mueller, Jr., and H. E. Smalley, Jr., "Control of a Partitionable Multimicroprocessor System," *1978 Int'l. Conf. on Parallel Processing*, Aug. 1978, pp. 9-17.

17. G. M. Masson, G. C. Gingher, and S. Nakamura, "A Sampler of Circuit Switching Networks," *Computer*, Vol. 12, No. 6, June 1979, pp. 32-48.

18. H. J. Siegel, R. J. McMillen, and P. T. Mueller, Jr., "A Survey of Interconnection Methods for Reconfigurable Parallel Processing Systems," *AFIPS Conf. Proc. 1979 NCC*, Vol. 48, June 1979, pp. 387-400.

19. K. J. Thurber, "Circuit Switching Technology: A State-of-the-Art Survey," *Proc. COMPCON 78* Fall, Sept. 1978, pp. 116-124.

20. K. J. Thurber, "Parallel Processor Architectures —Part 1: General Purpose Systems," *Computer Design*, Vol. 18, No. 1, Jan. 1979, pp. 89-97.

21. M. J. Flynn, "Very High-Speed Computing Systems," *Proc. IEEE*, Vol. 54, Dec. 1966, pp. 1901-1909.

22. D. J. Kuck, "A Survey of Parallel Machine Organization and Programming," *ACM Computing Surveys*, Vol. 9, Mar. 1977, pp. 29-59.

23. G. J. Lipovski and K. L. Doty, "Developments and Directions in Computer Architecture," *Computer*, Vol. 11, No. 8, Aug. 1978, pp. 54-67.

24. K. J. Thurber, *Large Scale Computer Architecture: Parallel and Associative Processors*, Hayden Book Company, Inc., Rochelle Park, N.J., 1976.

25. K. J. Thurber and L. D. Wald, "Associative and Parallel Processors," *ACM Computing Surveys*, Vol. 7, Dec. 1975, pp. 215-255.

26. D. Lawrie, "Access and Alignment of Data in an Array Processor," *IEEE Trans. Comput.*, Vol. C-24, No. 12, Dec. 1975, pp. 1145-1155.

27. K. E. Batcher, "The Flip Network in STARAN," *1976 Int'l. Conf. on Parallel Processing*, Aug. 1976, pp. 65-71.

28. H. J. Siegel, "Preliminary Design of a Versatile Parallel Image Processing System," *3rd Biennial Conf. on Computing in Indiana*, Apr. 1978, pp. 11-25.

29. H. J. Siegel and S. D. Smith, "Study of Multistage SIMD Interconnection Networks," *5th Annual Symp. on Computer Architecture*, Apr. 1978, pp. 223-229.

30. K. E. Batcher, "The Multidimensional Access Memory in STARAN," *IEEE Trans. Comput.*, Vol. C-26, No. 2, Feb. 1977, pp. 174-177.

31. L. H. Bauer, "Implementation of Data Manipulating Functions on the STARAN Associative Processor," *1974 Sagamore Computer Conf. on Parallel Processing*, Aug. 1974, pp 209-227.

32. S. W. Golomb, "Permutations by Cutting and Shuffling," *SIAM Review*, Vol. 3, Oct. 1961, pp. 293-297.

33. P. B. Johnson, "Congruences and Card Shuffling," *American Mathematical Monthly*, Vol. 63, Dec. 1956, pp 718-719.

34. H. S. Stone, "Parallel Processing with the Perfect Shuffle," *IEEE Trans. Comput.*, Vol. C-20, No. 2, Feb. 1971, pp. 153-161.

35. T. Lang and H. S. Stone, "A Shuffle-Exchange Network with Simplified Control," *IEEE Trans. Comput.*, Vol. C-25, No. 1, Jan. 1976, pp. 55-65.

36. T. Feng, "Data Manipulating Functions in Parallel Processors and Their Implementations," *IEEE Trans. Comput.*, Vol. C-23, No. 3, Mar. 1974, pp. 309-318.

37. S. D. Smith and H. J. Siegel, "Recirculating, Pipelined, and Multistage SIMD Interconnection Networks," *1978 Int'l. Conf. on Parallel Processing*, Aug. 1978, pp. 206-214.

38. G. R. Goke and G. J. Lipovski, "Banyon Networks for Partitioning Multiprocessor Systems," *1st Annual Symp. on Computer Architecture*, Dec. 1973, pp. 21-28.

39. S. D. Smith and H. J. Siegel, "An Emulator Network for SIMD Machine Interconnection Networks," *6th Annual Int'l. Symp. on Computer Architecture*, Apr. 1979, pp. 232-241.

40. C. Wu and T. Feng, "Routing Techniques for a Class of Multistage Interconnection Networks," *1978 Int'l. Conf. on Parallel Processing*, Aug. 1978, pp. 197-205.

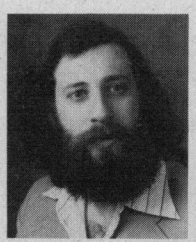
41. S. E. Orcutt, "Implementation of Permutation Functions in Illiac IV-Type Computers," *IEEE Trans. Comput.*, Vol. C-25, No. 9, Sept. 1976, pp. 929-936.

42. H. J. Siegel, "Analysis Techniques for SIMD Machine Interconnection Networks and the Effects of Processor Address Masks," *IEEE Trans. Comput.*, Vol. C-26, No. 2, Feb. 1977, pp. 153-161.

43. H. J. Siegel, "Single Instruction Stream-Multiple Data Stream Machine Interconnection Network Design," *1976 Int'l. Conf. on Parallel Processing*, Aug. 1976, pp. 272-282.

44. H. J. Siegel, "Partitionable SIMD Computer System Interconnection Network Universality," *16th Annual Allerton Conf. on Communication, Control, and Computing*, Oct. 1978, pp. 586-595.

45. T. Lang, "Interconnections Between Processors and Memory Modules Using the Shuffle-Exchange Network," *IEEE Trans. Comput.*, Vol. C-26, No. 5, May 1976, pp. 496-503.



Howard Jay Siegel is an assistant professor in the School of Electrical Engineering at Purdue University and on the research staff of Purdue's Laboratory for Applications of Remote Sensing. His research interests include parallel processing, multimicroprocessor systems, speech processing, natural language processing, and image processing. His current activities include designing a partitionable SIMD/MIMD multimicroprocessor system for image processing.

Siegel received SB degrees in electrical engineering and in management in 1972 from MIT. He received the MA and MSE degrees in 1974, and the PhD degree in 1977, all from Princeton University. He is a member of ACM and IEEE, and chairman of the Central Indiana Chapter of the IEEE Computer Society. He is a member of Eta Kappa Nu and Sigma Xi honorary societies.