STUDY OF MULTISTAGE SIMD INTERCONNECTION NETWORKS

Howard Jay Siegel and S. Diane Smith
Purdue University, Electrical Engineering School
West Lafayette, Indiana 47907

Abstract

Four SIMD multistage networks - Feng's data manipulator, STARAN flip network, omega network, and indirect binary n-cube -- are analyzed. Three parameters - topology, interchange box, and control structure -- are defined. It is shown that the latter three networks use equivalent topologies and differences in their capabilities result from the other parameters. An augmented data manipulator network using a modified control structure to perform more single pass interconnections than the other networks is presented. Some problems may be solved more efficiently if the 2^n processing elements of an SIMD machine can be partitioned into submachines of size 2^r. Single and multiple control partitioning are defined. The capabilities of these multistage networks to perform in these partioned environments are discussed.

Introduction

The age of the microprocessor has made feasible large-scale multiprocessor computer systems with as many as 2^{14} to 2^{16} processors [10,14]. The SIMD (single instruction stream - multiple data stream) [6] mode of parallel processing provides a structure for multimicroprocessor systems that is capable of performing a variety of tasks from matrix multiplication to image processing. Typically an SIMD machine has a control unit which broadcasts instructions to $N=2^n$ processors, numbered (addressed) from 0 to N-1, and all active processors execute the same instruction at the same time (for a detailed model of SIMD machines see [12]). The interconnection network can be used to connect processors to memory modules. Alternatively, the network can be used to interconnect processing elements, where each processing element (PE) consists of a processor and a memory module.

Three parameters for characterizing multistage interconnection networks are defined: topology, interchange box, and control structure. Two types of SIMD machine partitioning are introduced: single control and multiple control. These five features will be used to analyze and compare four multistage networks that have been proposed in the literature: Feng's data manipulator [5], Batcher's STARAN flip network [2], Lawrie's omega network [8], and Pease's indirect binary n-cube [10]. For each network numerous interconnection capabilities that make the network very useful have been demonstrated [1-5,8,10]. The analyses presented here will allow an architect to determine which parameters are required for a network to perform the tasks the system is being designed to handle. The basic connection pattern underlying many of these SIMD multistage networks has also been suggested for MIMD (multiple instruction stream - multiple data stream) [6] machines [9,14].

This work was supported in part by the Purdue Research Foundation under an XL Grant and a David Ross Grant (PRF#8718).

Multistage Network Parameters

The topology of the network is the actual interconnection pattern used to connect the set of input lines and output lines. The generalized cube topology is

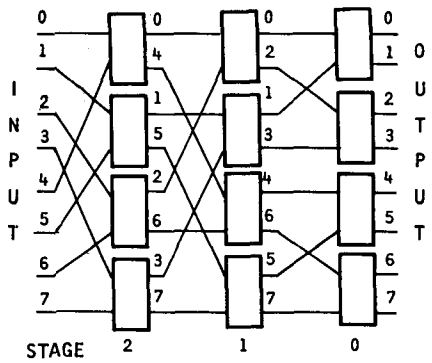

Figure 1: 8-item generalized cube network.

defined to consist of n stages, where $N=2^n$ is the number of input and output lines, as shown in Figure 1. Each stage consists of an interconnection pattern of N lines and N/2 interchange boxes. For each box the upper and lower outputs are labeled with the same numbers as the upper and lower inputs, respectively. At stage i the input lines that differ only in their i-th bit position are paired together as inputs to an interchange box, for $0 \leq i < n$. The data first passes through stage n-1, then n-2, 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. The fundamental connection patterns which form the basis for the generalized cube topology are discussed in [11-13].

The interchange box is a two-input two-ouput device. 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 [8]. There are also two illegitimate states: (1) lower conflict - inputs i and j both to output j; and (2) upper conflict - inputs i and j both to output i. Both of these states involve conflicts, the condition when both inputs attempt to feed the same output. A two-function interchange box is an interchange box capable of being in either the straight or exchange states. A four-function interchange box is an interchange box capable of being in any of the four legitimate states.

The control structure of a network sets the states of the interchange boxes. 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. 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 Figure 2, the letters a, b, c,

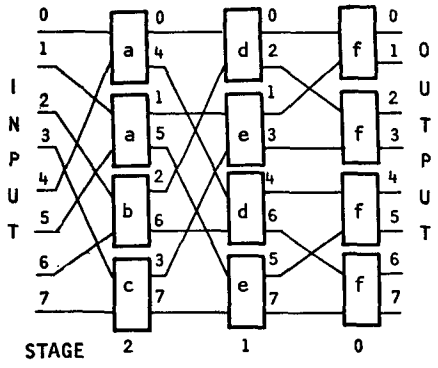

Figure 2: 8-item generalized cube network with partial stage control.

d, e, and f each represent a different control line. An interchange box is controlled by the line corresponding to its label. This is used for "shift" permutations as will be discussed later.

Comparisons of Network Parameters

The STARAN flip network [2] has N input lines and N output lines, both labeled from 0 to N-1. Input lines which differ in the i-th bit position are paired in the i-th stage of the network and the data passes through stage 0, then stage 1, etc., as shown in Figure 3. Thus, the flip network topology is identical to the generalized cube, except the order of the stages is reversed. The interchange boxes are the two-function type. The network has two control mechanisms, the flip control, generating "flip" permutations, and the shift control, generating "shift" permutations. An n-bit flip control vector is used for individual stage control. The shift control allows shifts of 2^m places modulo 2^P, $0 \leq m \leq P \leq n$, using i+1 control lines at stage i, $0 \leq i < n$, as shown in Figure 4. The flip control is implemented by controlling all the shift control lines for a given stage by a single signal.

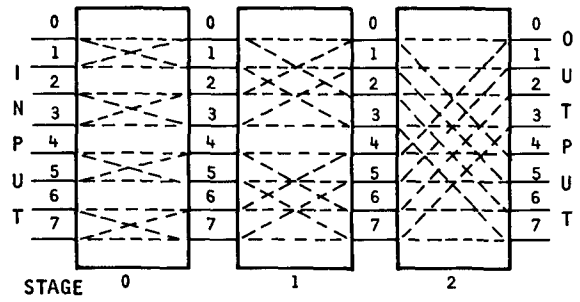

Figure 3: 8-item flip network with individual stage (flip) control [2].

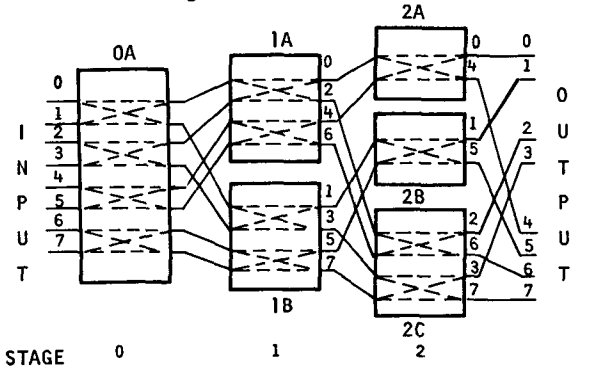

Figure 4: 8-item flip network with partial stage (shift) control [2]. 0A, 1A, etc. are the control signals.

Theorem: The flip network using flip control is functionally equivalent to the generalized cube network with two-function boxes using individual stage control.

Proof: Consider $I=i_{n-1}\ldots i_1 i_0$, an arbitrary input line to the flip network. If the flip control vector F equals $(f_{n-1}\ldots f_1 f_0)$, where f_i is 0 or 1, the data on input line I is sent to output line $I \oplus F = (i_{n-1} \oplus f_{n-1}, \ldots i_1 \oplus f_1, i_0 \oplus f_0)$, where $\oplus$ is the EXCLUSIVE-OR function [2]. This is because $f_i = 0$ sets stage i to straight and $f_i = 1$ sets it to exchange. Associate f_i with stage i of the generalized cube network. Then, input I will be sent to output $I \oplus F$ since no matter what input line data from a particular source is on when it passes through stage i it will exchange if $f_i = 1$ and will go straight through if $f_i = 0$ (this can be proven formally by induction). □

Theorem: The generalized cube network with two-function interchange boxes and partial stage control can perform the same 2^m shifts modulo 2^P, $0 \leq m \leq P \leq n$, that the flip network with shift control can perform.

	Control Signal					
Permutation	0A	1A	1B	2A	2B	2C
1 mod 8	1	1	0	1	0	0
2 mod 8	0	1	1	1	1	0
4 mod 8	0	0	0	1	1	1
1 mod 4	1	1	0	0	0	0
2 mod 4	0	1	1	0	0	0
1 mod 2	1	0	0	0	0	0
identity	0	0	0	0	0	0

Table 1: Shift control specifications for Figure 4 [2].

Proof: In Table 1, when a control signal is 1 the interchange boxes to which it is connected use the exchange state, otherwise they use the straight state. The partitioned stage control version of the generalized cube network can perform the same shifting tasks by equating the following pairs of control signals: f and 0A; d and 1B; e and 1A; a and 2C; b and 2B; and c and 2A (see Figures 2 and 4). This can be proven formally for arbitrary N based on the "Cube to PM2I" algorithm presented in [12]. □

It should be noted that the generalized cube network with two-function interchange boxes and partial stage control is not directly equivalent to the flip network; e.g., the generalized cube network can connect 7 to 0 and 5 to 6, while the flip network can not. This problem arises due to reversed order of the stages of the two networks, causing data that pass through different control groupings at stage i in one network to pass through the same control grouping at stage i in the other network.

These theorems prove that any network whose topology is functionally equivalent to the generalized cube and has two- or four-function interconnection boxes with partitioned stage or individual box control can perform all the "flip" and "shift" permutations which the flip network can achieve.

An N by N omega network, shown in Figure 5, 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 individu-

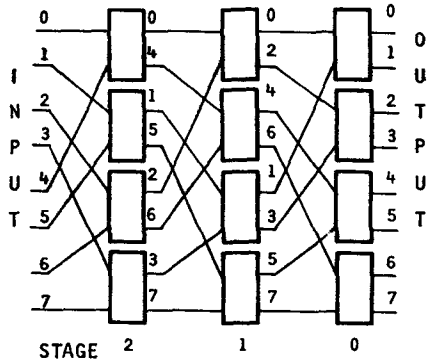

Figure 5: 8-item omega network [7].

al box control [7]. The shuffle connects output $p_{n-1}\cdots p_1p_0$ of stage i to input $p_{n-2}\cdots p_1p_0p_{n-1}$ of stage i-1. Each interchange box in an omega network is controlled by the n-bit destination tags associated with the data on its input lines.

Theorem: The omega network is functionally equivalent to the generalized cube network with four-function interchange boxes and individual box control.

Proof: As a result of the shuffle interconnection prior to each column of interchange boxes, at stage j two inputs are paired if and only if they differ only in the j-th bit position (this can be proved formally by induction). The data passes through the stages of the omega and generalized cube networks in the same order. □

Any network functionally equivalent to a generalized cube network with four-function boxes and individual box control can perform the same interconnection tasks the omega network can. Furthermore, any network functionally equivalent to a generalized cube network with two-function boxes and individual box control can perform the subset of these tasks that do not use data duplication, i.e., that do not use upper or lower broadcasts.

An indirect binary n-cube network consists of N/2 two-function interchange boxes under individual box control connected so that the two inputs to an interchange box at stage i differ only in the i-th bit position and data passes through stage 0, then stage 1, etc. as shown in Figure 6 [10]. This is the reverse of the order of the stages in the generalized cube network.

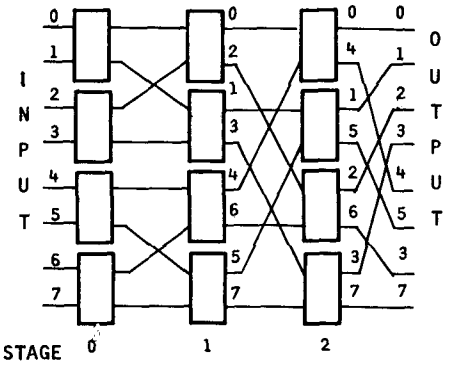

Figure 6: 8-item indirect binary n-cube network [9].

Theorem: The indirect binary n-cube network is functionally equivalent to the generalized cube network with two-function boxes and individual box control if address transformations are used.

Proof: To show that the two networks are not functionally equivalent without address transformations note that the n-cube can not connect 0 to 5 and 1 to 7 while the generalized cube can. This occurs because as data passes through these networks the data items that are paired in an interchange box depend on the order of the stages of the network.

Consider an arbitrary set of input and output connections which the n-cube can perform. To perform these connections on the generalized cube the following address transformations are necessary. Transform the number (address) of each input and output line from $p_{n-1}\cdots p_1p_0$ to $p_0p_1\cdots p_{n-1}$. Use this same transformation to specify any desired set of input and output connections.

For example, for N=8 the n-cube can perform generalized cube connections 0 to 5 and 1 to 7 by using the transformations, i.e., it can connect 0 to 5 and 4 to 7. The correctness of this address transformation method can be proven formally by realizing that the renumbering causes the generalized cube network to first pair data lines whose numbers differ only in the 0-th bit, then the 1-st bit, etc., just the way the n-cube does. (In [10] it is noted that the n-cube "is similar ... to the omega network itself with renumbering.") □

This theorem proves that any network functionally equivalent to a generalized cube network with two- or four-function boxes and individual box control can perform the same tasks the n-cube network can perform if an address transformation is used.

The data manipulator network consists of n stages of N cells, where for $0 \le j < N$ and $0 \le i < n$, there are three sets of interconnections going from cell j in stage i to stage i-1: to cell $j + 2^i \bmod N$, to cell $j - 2^i \bmod N$, and to cell j [5]. Each stage of the network is controlled by a pair of signals, specifying two of H_1, H_2, U_1, U_2, D_1, and D_2, as described in Figure 7. The topology is such that at

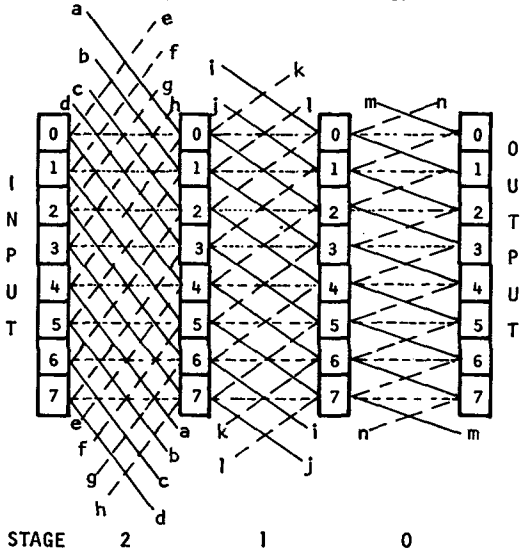

Figure 7: 8-item data manipulator network [5].

The dashed lines represent the U control line interconnections.
The dotted lines represent the H control line interconnections.
The solid lines represent the D control line interconnections.
For stage I, U_I, D_I, and H_I control those cells whose I-th bit is 0, and U_2, D_2, and H_2 control those cells whose i-th bit is 1.

stage i cell x is connected to cell x and to the cell which differs from x in only the i-th bit position, just as the topology of the generalized cube allows. In addition, the data manipulator connects x to both $x + 2^i \bmod N$ and $X - 2^i \bmod N$, one of which differs from x in more than the i-th bit position. The stages are ordered from n-1 to 0, as in the generalized cube topology.

An augmented data manipulator (ADM) is a data manipulator with individual cell control, i.e., each cell can get none, one, or two of the signals H, U, and D. The effects of these controls are as described in Figure 7 for the regular data manipulator, except that each cell can be controlled independently.

Theorem: The ADM can perform all the interconnections the generalized cube with four-function boxes and individual box control can achieve and some that the generalized cube can not achieve.

Proof: Consider simulating the four-function box at stage i whose upper input is x and lower input is y. Let the subscript x denote the ADM control signal for cell x at stage i, and let the subscript y be used similarly. Then to perform the four functions use the following control signals:

straight - H_xH_y, exchange - D_xU_y,

lower broadcast - U_yH_y, and

upper broadcast - D_xH_x.

In addition, the ADM network can perform interconnections the generalized cube cannot, such as connecting 3 to 3, 5 to 2, and 6 to 6, for N=8. □

Partitioning

Some computations may be more efficiently executed if the N PE's of the SIMD machine can be partitioned into 2^{n-r} groups of 2^r PE's. Each group may then behave like a smaller SIMD computer, and the system resources may be more efficiently utilized. As an example, consider an algorithm that is composed of k independent subalgorithms, each of which can be executed using 2^r or fewer PE's. Suppose that each subalgorithm requires at most j time periods to execute. The algorithm could be performed one subalgorithm at a time, using at most 2^r PE's at any time. The computation would require at most j*k time periods, and $2^n - 2^r$ PE's would be unused. If on the other hand, the 2^n PE's were partitioned into 2^{n-r} groups of 2^r PE's, 2^{n-r} of the k subalgorithms could be performed in parallel (assuming that all groups use the same instruction stream or that multiple control units are available). Now, the computer is more fully utilized, and if $k \leq 2^{n-r}$, the algorithm can be executed in j time periods.

The number of PE's that can be utilized for a given computation is highly algorithm dependent. Two classes of partitioning for an SIMD machine will be considered: (1) use the system as one to 2^{n-r} SIMD machines, each having 2^r PE's, each performing the same algorithm (using the same instruction stream), but each on a different data set; and (2) use the system as one to 2^{n-r} SIMD machines, each of size 2^r, where each may be performing a different algorithm on a different data set. The first class will be referred to as single control partitioning, since only one control mechanism will serve all of the PE's. The second class will be referred to as multiple control partitioning, since a separate control mechanism will be required for each instruction stream. Note that multiple control partitioning includes having, for example, four groups following one instruction stream, while four other groups each follow their own instruction streams. Furthermore, it may be the case that two or more groups may be executing different SIMD programs on the same logical data set, although each group would have its own physical copy. Note that the single control class is a special case of the multiple control class, where all the groups use copies of the same instruction stream.

When the machine is partitioned, each submachine needs an interconnection network. An interconnection network may be described as a set of interconnection functions, where each interconnection function is a permutation (bijection) on the set of PE addresses [11]. If f is an interconnection function, then "f(x) = y" means that the function connects input x to output y.

An n stage shuffle-exchange network, such as the omega network [8] shown in Figure 5, can be defined with two interconnection functions. For N PE's, let the binary representation of a PE address P be $P = p_{n-1} \cdots p_1p_0$. Then, the perfect shuffle is defined as [11]

$$\text{Shuffle}(p_{n-1}p_{n-2}\cdots p_1p_0) = p_{n-2}\cdots p_1p_0p_{n-1}.$$

The exchange function is defined as [11]

$$\text{Exchange}(p_{n-1}p_{n-2}\cdots p_1p_0) = p_{n-1}p_{n-2}\cdots p_1\bar{p}_0,$$

where $\bar{p}_0$ denotes the complement of p_0. Each stage of the network consists of a shuffle and an optional exchange.

Partitioning a multistage shuffle-exchange network is discussed in [7]. That discussion is extended here and is related to the two types of partitioning defined above.

Theorem: Let the system be partitioned into 2^{n-r} subsystems of 2^r PE's each. If only a shuffle and no exchange is allowed on each of the first n-r stages of the network, the network can serve as a complete r stage shuffle-exchange network for each of the 2^{n-r} groups of PE's of the partitioned machine.

Proof: Let the addresses of all PE's in each partition have the same n-r most significant bits. Let loc(i)(P) refer to the location of the data originally in PE(P) after passing through i stages of the network. After n-r shuffles with no exchanges, the intermediate address of the data that was initially in P is

$$\begin{aligned}\text{loc}(n-r)(P) &= p_{n-(n-r)-1}\cdots p_1p_0p_{n-1}\cdots p_{n-(n-r)} \\ &= p_{r-1}\cdots p_1p_0p_{n-1}\cdots p_r.\end{aligned}$$

Lastly, r shuffle-exchange steps follow, and the final address of the data is

$$\text{loc}(n)(P) = p_{n-1}\cdots p_rd_{r-1}\cdots d_1d_0,$$

where $d_i = \bar{p}_i$ or p_i. All PE's with addresses that have the same n-r most significant bits will be in the same partition of the machine.

Analogous arguments may be made to show that the PE's may be partitioned based on any common set of n-r bits of the PE addresses. □

If the shuffle-exchange network employs individual box control, it is equally applicable for both single control partitioning and multiple control partitioning. Consider a 3 stage shuffle-exchange network that is to be partioned into two groups of four PE's each. Let the two groups be G1 = (0, 1, 2, 3) and G2 = (4, 5, 6, 7).

Consider the case where only G1 is using the network. Let c be a control matrix for the network of Figure 5. c(i,j) controls the i-th row exchange module at stage (n-1)-j. An exchange occurs at that modu-

le if and only if c(i,j) = 1. To control the network for G1, the matrix is

$$c = \begin{bmatrix} 0 & c(0,1) & c(0,2) \\ 0 & - & c(1,2) \\ 0 & c(2,1) & - \\ 0 & - & - \end{bmatrix}$$

Note that only the control bits that relate to the submachine that is using the network need be specified. All others may be ignored. To control the network for G2 alone, the matrix is

$$c = \begin{bmatrix} 0 & - & - \\ 0 & c(1,1) & - \\ 0 & - & c(2,2) \\ 0 & c(3,1) & c(3,2) \end{bmatrix}$$

If G1 and G2 are to share the network under a single control partitioning scheme, set c(0,1) = c(1,1), c(2,1) = c(3,1), c(0,2) = c(2,2), and c(1,2) = c(3,2).

For multiple control partitioning, G1 is controlled by c(0,1), c(2,1), c(0,2), and c(1,2), while G2 is controlled by c(1,1), c(3,1), c(2,2), and c(3,2). Thus, if c(i,0)=0, $0 \leq i < 4$, the two partitions can operate independently and asynchronously.

Theorem: An n stage shuffle-exchange network with individual box control c(i,j) may be used with multiple or single control partitioning.

Proof: Divide the 2^n PE's into 2^{n-r} groups of 2^r and that the PE addresses of a group G are consecutive and range from 2^k to $2^k + 2^r - 1$, where $2^k = j2^r$, j is an integer, $0 \leq j < 2^{n-r}$. The n-r most significant bits, $p_{n-1} \cdots p_r$, of all PE's in G are the same. The first n-r stages allow only a shuffle and no exchange, so the entries of the first n-r columns of c are all 0, and

$$\text{loc } (n-r)\ (P) = p_{r-1} \cdots p_1 p_0 p_{n-1} \cdots p_r.$$

Stage r-1 is the first to allow an exchange to follow the shuffle. Let c(A, n-r) be a control bit for stage r-1 where $A = a_{n-2}a_{n-3} \cdots a_1 a_0$, since there are 2^{n-1} rows of c. Then c(A, n-r) controls PE's belonging to G if

$$A = a_{n-2} \cdots a_{n-r} p_{n-1} \cdots p_r.$$

At stage r-(1+j), $1 \leq j < r$, c(A, n-r+j) controls PE's of G if

$$A = a_{n-2} \cdots a_{n-r+j} p_{n-1} \cdots p_r a_{j-1} \cdots a_0.$$

For single and multiple control partitioning, if the group of PE's is known, the control signals that refer to that group are given by the above expression. □

A disadvantage of the scheme is that the c matrix requires $n \times 2^{n-1}$ storage locations, which may become prohibitive, both for storage and programming. In [7] an approach is presented that allows each exchange box to calculate its own control. However, this network can not perform all the permutations of a shuffle-exchange network. The omega network [8] passes destination tags, thus effectively passing each row of the c matrix with the data.

As an alternative to the multistage implementation of the shuffle-exchange, a "recirculating" implementation may be used [12,13]. This would allow each PE to store one row, n bits, of the control matrix or to compute its own control bits based on its address or the data it contains.

In the STARAN flip network (Figure 3), stage i can perform the cube-type interconnection function c(i), where

$$c(i)\ (P) = p_{n-1}p_{n-2} \cdots p_{i+1}\bar{p}_i p_{i-1} \cdots p_1 p_0,$$

for $0 \leq i < n$ [11].

The shuffle-exchange and the flip network have equivalent topologies, and the partitioning proofs are similar.

Theorem: The STARAN flip network may be partitioned under single control partitioning.

Proof: The control vector F of the flip network allows the partitioning of the network. In order to partition the machine into 2^{n-r} groups of 2^r PE's, where the addresses in each group have the same n-r most significant bits, the control vector used is

$$F = (\ 0 \ \ldots\ 0\ f_{r-1} \cdots f_1\ f_0\ ).$$

For an arbitrary PE address, P, the first r stages of the network produce

$$\text{loc}(r)\ (P) = p_{n-1} \cdots p_r d_{r-1} \cdots d_0,$$

where $d_i = \bar{p}_i$ or p_i. If the last n-r stages allow no exchanges, the address of the final destination of the data is

$$\text{loc}(n)\ (P) = p_{n-1} \cdots p_r d_{r-1} \cdots d_0.$$

Thus, all PE's with the same n-r most significant bits will be in the same partition of the reconfigurable machine. Again, this procedure can be generalized to group PE's that have any set of n-r bits in common. □

For single control partitioning, the individual stage control of the flip network negates the need for equating control bits as in the shuffle-exchange case. Thus, for single control partitioning, the flip network can act as up to 2^{n-r} flip networks of size 2^r.

If up to 2^{n-r} groups of PE's have access to the network at the same time to do different tasks, as in multiple control partitioning, then the control structure is not general enough. The flip network shift control cannot be effectively used for either type of partitioning.

Theorem: The indirect binary n-cube with individual box control can be partitioned under single or multiple control partitioning.

Proof: The proof is analogous to that for the shuffle-exchange network, since the two networks have the same topologies. The only difference between the two cases is the positioning of the control matrix entries which control each position. □

The data manipulator network is based on the 2n "PM2I" interconnection functions

$$t_{+i}\ (j) = (j + 2^i) \bmod N$$

$$t_{-i}\ (j) = (j - 2^i) \bmod N$$

for $0 \leq j < N$, $0 \leq i < n$ [11]. The data manipulator is an n stage network where stage i can perform t_{+i} (i.e., D), t_{-i} (i.e., U), and no change (i.e., H), as shown in Figure 7. Recall that for stage 2^i, U_1, D_1, and H_1 control those PE's whose i-th bit is 0, and U_2, D_2, and H_2 control those PE's whose i-th bit is 1.

Theorem: The data manipulator network may be divided into 2^{n-1} groups of 2 adjacent PE's for either single or multiple control partitioning.

Proof: Let the two PE's be PE(i) and PE(i+1 mod N), for i even. Since N = 2, the network need consist only of $t_{\pm 0}$ (note that for N = 2, t_{+0} is the same as t_{-0}). This can be obtained using D_1U_2 on stage 0 and $H_1\ H_2$ on all other stages. Similarly, PE(i) and PE(i+1 mod N), for i odd, can exchange data if $U_1\ D_2$ is used on stage 0 and $H_1\ H_2$ is used on all other stages. This is true for multiple control partitioning since PE's not exchanging data can ignore the network. □

As an additional point, consider pairing two arbitrary PE's.

Theorem: For the data manipulator network, for the control functions given as in Figure 7, if only two PE's use the network at one time, then any two PE's can exchange data in one pass through the network.
Proof: For N PE's, let PE(A) and PE(B) exchange data, where A = a(n-1)...a(0), B = b(n-1)...b(0), and A ≠ B. In order for PE(A) and PE(B) to exchange data, transform the addresses such that a(n-1)'...a(0)' = b(n-1)...b(0) and b(n-1)'...b(0)' = a(n-1)...a(0). An algorithm to do so is as follows.
For stage i, i = n-1, n-2, ... 0:
If a(i) = b(i) , then no change (H_1 H_2).
If a(i) = 0 and b(i) = 1, then apply t_{+i} to the data from A, and apply t_{-i} to the data from B (D_1 U_2). Thus,
a(i)' = a(i) + 1 = 1 = b(i), and
b(i)' = b(i) - 1 = 0 = a(i).
If a(i) = 1 and b(i) = 0, then apply t_{-i} to the data from A, and apply t_{+i} to the data from B (D_1 U_2). Thus,
a(i)' = a(i) - 1 = 0 = b(i), and
b(i)' = b(i) + 1 = 1 = a(i).
After the i-th stage,
loc(i) (A) = b(n-1)...b(n-i)a(n-i-1)...a(0), and
loc(i) (B) = a(n-1)...a(n-i)b(n-i-1)...b(0).
Finally, after n stages,
loc(n) (A) = b(n-1)...b(0), and
loc(n) (B) = a(n-1)...a(0). □

This algorithm may be adapted for the n stage shuffle-exchange, the indirect binary n-cube, and the flip networks. For the shuffle-exchange, if bit i of the two PE addresses, A and B, are the same, no exchange occurs at stage i, and c(A, n-1-i) = 0, $0 \leq A < 2^{n-1}$. If the two bits differ, then an exchange occurs, and c(A, n-1-i) = 1, $0 \leq A < 2^{n-1}$. For the flip network, at stage i, if the i-th bits of A and B are the same, then $f_i = 0$, and no exchange occurs. If the two bits differ, then $f_i = 1$, and an exchange occurs. The algorithm for the n-cube is similar. For the algorithm, the data manipulator has behaved as a generalized cube network, so the similarities of the algorithms are not unexpected.

If a restriction is made, the data manipulator network can be partitioned such that each group has a complete data manipulator network at its disposal. For N = 8, let the machine be partitioned into 2^1 groups of 2^2 PE's. Let one group contain PE's 0, 2, 4, and 6, while the other group contains PE's 1, 3, 5, and 7. The control structure of this network is not general enough to allow multiple control paritioning, and so only single control partitioning will be considered. If only the straight through path (H_1 H_2) is allowed at stage 0, then each group of 4 PE's has a $\log_2 4$ stage data manipulator network available to it. The 1 level now corresponds to a 0 level for each partitioned group, since each PE address is two apart from its neighbor's addresses. t_{+1} moves data from position 0 to position 2, from 2 to 4, from 4 to 6, and from 6 to 0. Similarly, the 2 stage corresponds to a 1 stage for the group. Figure 8 illustrates this process.

Theorem: Let the data manipulator be partitioned into 2^{n-r} groups of 2^r PE's so that all PE's in a partition have the same n-r low order bits. Under this restriction, the data manipulator may be partitioned under single control partitioning.

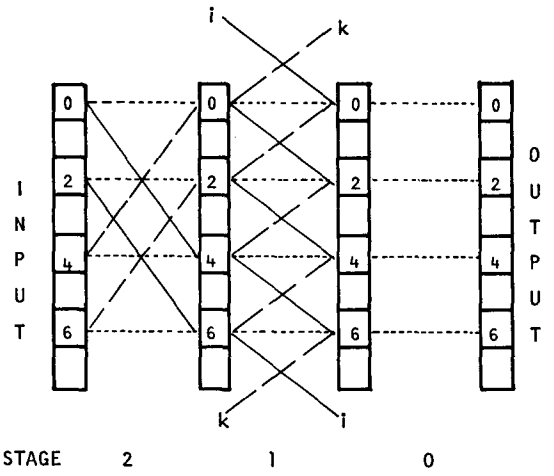

Figure 8: 8-item data manipulator network for the use of four PE's only (0,2,4,and 6). Note that for N=8 $t_{+2}=t_{-2}$ (i.e., U=D).

Proof: Each group is composed such that consecutive PE's within the group have PE addresses that differ by 2^{n-r}. Then stage n-r behaves as a stage 0 for the group, stage (n-r+1) behaves as a stage 1, and so on, and lastly, stage n-1 behaves as a stage (r-1) for the group of 2^r PE's. □

While this method does group 2^r PE's together, the PE addresses are not consecutive, and this may cause confusion for the programmer. A "logical" t_{+0} interconnection for the group 0, 2^{n-r},..., $(2^r - 1)2^{n-r}$ means to send data from 0 to 2^{n-r}, 2^{n-r} to $2^{n-r}2$, and so forth. It does not mean to send data from 0 to 1, etc. Also, this logical t_{+0} interconnection is a "physical" $t_{+(n-r)}$ interconnection on the network.

There are means of allowing 2^r PE's with consecutive addresses to be grouped together for this network and still have a complete network available. However, problems arise due to missing connections, such as the 0 to $2^r - 1$ connection for t_{+0}.

In summary, for multiple control partitioning, the data manipulator network is adequate to divide the PE's into 2^{n-1} groups of two PE's where a group consists of PE(i) and PE(i+1 mod N), $0 \leq i < N$. With single control partitioning, certain sets of 2^r PE's may have an r stage network available, but the grouping of 2^r PE's is more restricted than with the indirect binary n-cube, the flip, and shuffle-exchange networks. Like the flip network, the control structure here is not general enough to allow multiple control partitioning. The ADM network has the additional control capability to allow multiple control.

Conclusions

Three parameters for classifying multistage SIMD machine interconnection networks have been presented: topology, interchange box, and control structure. The generalized cube topology has been defined and used as a standard to which to compare other multistage networks. Six states for interchange box functions were described and three types of control schemes were defined.

It was shown that the STARAN flip network, the indirect binary n-cube network, and the omega network all use topologies equivalent to the generalized cube. The omega network sends its data through stage n-1, then n-2, etc., while the other two networks use the reverse order. The omega network uses four-function interchange boxes, while the other two networks use two-function boxes. The flip network uses partial stage control and the other two networks use individual box control. The networks may be ordered in terms of increasing capability as follows: flip network, indirect binary n-cube, and omega network. Due to the reversed order of the stages, the omega network may have to use address transformations to perform all of the n-cube interconnections. One possible alternative to address transformations is to construct the network so that it is bidirectional.

Thus, a system architect beginning with the generalized cube topology can add features to improve the flexibility of the system network. The information presented about the flip, n-cube and omega networks in [1-3,7,9] can be used in conjunction with the information presented here to do a cost-performance analysis based on the intended uses of the architecture being designed.

The augmented data manipulator (ADM) network was introduced. It is based on the data manipulator network, but it uses a more flexible control scheme. It was shown that this network can perform any interconnection task the generalized cube network with four-function boxes and individual box control can perform (hence, any task the flip, omega, or n-cube can perform). In addition, it was shown that the ADM network can perform tasks the generalized cube can not (hence, tasks the flip, omega, and n-cube networks can not perform). As in the case of choosing features for a generalized cube network, a system architect must decide if the added flexibility of the ADM network is worth the additional cost to construct it.

Two ways to partition an SIMD machine consisting of 2^n processing elements into two to 2^{n-r} groups of 2^r PE's, each group acting as an SIMD machine of size 2^r have been defined. One way, single control partitioning, used a single control to broadcast instructions to all groups. Thus, the same SIMD machine program may be executed on several different data sets simultaneously (e.g., multiplying two pairs of matrices). The other way, multiple control partitioning, assumes the availability of additional control units so that each 2^r PE submachine may have its own instruction stream. The data being acted on by different submachines may be copies of the same data file (e.g., processing two copies of the same image to find different features).

The way in which various types of networks may function in both of these partitioning environments was examined. Under single control partitioning, the shuffle-exchange, n-cube, and flip networks are readily partitioned into 2^{n-r} groups of 2^r PE's. The data manipulator network can group certain sets of 2^r PE's, but the ability to connect sets of 2^r PE's with consecutive addresses is restricted.

The flip and data manipulator networks do not have a control structure that is general enough to accommodate multiple control partitioning. The omega, n-cube, and ADM network have control structures which allow them to function in a multiple control environment.

References

[1] K. E. Batcher, "The multidimensional access memory in STARAN," IEEE Trans. Comput., Vol. C-26 (Feb., 1977), pp. 174-177.

[2] K. E. Batcher, "The flip network in STARAN," 1976 Int. Conf. on Parallel Processing, (Aug. 1976), pp. 65-71.

[3] 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.

[4] T. Feng, Parallel Processing Characteristics and Implementation of Data Manipulating Functions, Dept. of Electrical and Computer Engineering, Syracuse University, RADC-TR-73-189 (Jul., 1973).

[5] T. Feng, "Data manipulating functions in parallel processors and their implementations," IEEE Trans. Comput., Vol. C-23 (Mar., 1974), pp. 309-318.

[6] M. J. Flynn, "Very high-speed computing systems," Proceedings of the IEEE, Vol. 54 (Dec., 1966), pp. 1901-1909.

[7] T. Lang and H. S. Stone, "A shuffle-exchange network with simplified control," IEEE Trans. Comput., Vol. C-25 (Jan., 1976), pp. 55-65.

[8] D. Lawrie, "Access and alignment of data in an array processor," IEEE Trans. Comput., Vol. C-24 (Dec., 1975), pp. 1145-1155.

[9] G. J. Lipovski and A. Tripathi, "A reconfigurable varistructure array processor," 1977 Int. Conf. on Parallel Processing (Aug., 1977), pp. 165-174.

[10] M. C. Pease, "The indirect binary n-cube microprocessor array," IEEE Trans. Comput., Vol. C-26 (May, 1977), pp. 458-473.

[11] H. J. Siegel, "Analysis techniques for SIMD machine interconnection networks and the effects of processor address masks," IEEE Trans. Comput., Vol. C-26 (Feb., 1977), pp. 153-161.

[12] H. J. Siegel, "Single instruction stream - multiple data stream machine interconnection network design," 1976 Int. Conf. on Parallel Processing (Aug., 1976), pp. 272-282.

[13] H. J. Siegel, "The universality of various types of SIMD machine interconnection networks," Fourth Annual Symposium on Computer Architecture (Mar., 1977), pp. 70-79.

[14] H. Sullivan, T. R. Bashkow, and K. Klappholz, "A large scale homogeneous, fully distributed parallel machine," Fourth Annual Symposium on Computer Architecture (Mar., 1977), pp. 105-124.