Interconnection networks—A survey and assessment

by KENNETH J. THURBER
Sperry Univac
St. Paul, Minnesota

INTRODUCTION TO INTERCONNECTION NETWORKS

As the level of complexity of digital systems increases, the problem of interconnecting subunits is receiving increasing attention. We are reaching the point where processing speed cannot be further improved through the use of faster componentry. Further speed-up of systems will most likely result from changes in the organization and structure of hardware, rather than by raw circuit improvements. Another factor increasing the complexity of systems is the arrival of cheap, powerful LSI microcomputers which allow system construction involving a plurality of processors connected together to perform a specific task. Restructurable system concepts are also very promising, but require extensive amounts of interconnective capability. Thus, bus structures are attracting considerable attention. This paper focuses on a small segment of the general bus structure problem; namely, interconnection (permutation, sorting, etc.) networks.

Figure 1 indicates the essential elements of an interconnection network. The network consists of a set of \( n \) input lines, a set of \( n \) output lines, a block of connective logic, and a set of control lines. The control lines structure the connective logic such that the \( n \) input lines and \( n \) output lines are connected together in some fashion. Clearly, if we allow all permutations of the line connections to occur there are \( n! \) connections possible. Thus, an upper bound on the network control lines required is \( \log_2(n!) \). There are a number of variations of such networks that have been recently proposed. This paper will describe each of the major concepts and will give a global comparison of the concepts as to their capability, throughput delay, allowable size, etc.

Typical applications of the networks will also be indicated. One interesting point to note is that since the functionality of the networks is so difficult to realize, little work has been done on the networks to make them practical other than trying to discover ways to cost effectively achieve their functionality.

Sorting via software techniques has been studied extensively. Martin presents a comprehensive survey of the software sorting problem and its various solutions.

DESCRIPTION OF INTERCONNECTION NETWORK CONCEPTS

The most important interconnection network concepts will be described in this section in chronological order.

Benes' telephone network

Benes may be considered the father of interconnection networks. He performed the first major in depth studies of the rearrangeable array and the two input-two output permuter cell. Additionally, he studied the networks in terms of their combinatorial and topological properties.

Benes defined a connecting network as an arrangement of switches and disjointed transmission links which allow a set of terminals to be connected together in various combinations. Benes also studied the rearrangeability of a network and the concepts of "blocking" and "distance." Rearrangeability deals with the ability to route new calls during the presence of current calls. A network is considered blocked, if a given pair of idle terminals cannot be connected. A network can be non-blocking in two senses. These are: (1) if rearrangement of present calls unblocks the requested call (non-blocking in the wide sense) or (2) if a network has no blocking states (non-blocking in the strict sense). Distance is the number of calls one would have to add or remove to change network states.

Benes uses graphs to describe his networks and derives necessary and sufficient conditions for them to be non-blocking in both senses.

A diagram of a typical rearrangeable network is shown in Figure 2. A Benes network for \( n=8 \) is given in Figure 3. Further extensive work on rearrangeable networks has been done by Joel and Opferman and Tsao-Wu. Opferman and Tsao-Wu indicate that extensions to \( n \times n \) networks can be readily made using the rearrangeable array.

Batcher networks

Batcher networks were envisioned for use in sorting and merging applications and as a replacement for crossbar switching networks. A crossbar switch for \( n \) elements grows at the rate of \( n^2 \). A Batcher network grows at the rate of \( (\frac{1}{2})n(\log_2 n)^2 \) for \( n \) elements. The delay time for \( n=2^p \) words is \( (\frac{1}{2})p(p+1) \) basic element delays; i.e., \( (\frac{1}{2}) \log_2 n \) \((\log_2 n) +1\).

Figure 4 shows the basic Batcher network element. It accepts two input numbers \( A \) and \( B \) and outputs their minimum (maximum) on the \( L (H) \) output line. Figure 5 is the symbol for a \( s \) by \( t \) odd-even merging network. The use of the comparison network to merge two ascending sorted lists \( (a_1, \ldots, \)
and $b_1, \ldots, b_n$) into a single ascending sorted list $(c_1, \ldots, c_{n+1})$ is shown in Figure 6. Since a 1 by 1 network is a simple comparator, Figure 6 is an iterative rule for the odd-even merging network's construction.

Another network (Bitonic Sorter) was also described. A sequence is defined as bitonic if it is the juxtaposition of two monotonic sequences, one ascending and one descending. Also it can be assumed that a sequence remains bitonic if it is split and the two parts are interchanged. Since any two monotonic sequences can be used to construct a bitonic sequence, a network that rearranges a bitonic sequence into monotonic order can be built and used as a merging network. Figure 7 gives the iterative rule for constructing a bitonic sorting network for $2n$ numbers using $n$ comparison cells and two sorters for $n$ numbers.

Sorting networks for arbitrary sequences can be constructed from either odd-even or bitonic sorters by forming ordered lists of length 2 and merging those lists two at a time, to form lists of length 4, etc.

---

**Figure 1—An interconnection network**

**Figure 2—Benes' rearrangeable network**

**Figure 3—An 8×8 rearrangeable network**

**Figure 4—Basic batcher cell**
Much work was done on interconnection networks at Stanford Research Institute. Kautz provides a good summary of the results of the various studies. He describes three main problems. These are: (1) universality of interconnection networks and (2) minimization of interconnection networks and (3) permutation networks.

Problems (1) and (2) deal with the desire to be able to arrange a homogeneous set of elements in a "less-than-completely-connected" network to optimize some parameter. The universality of interconnection networks is studied by using networks represented as linear graphs. Problem (1) becomes the determination of the characteristics of universal networks which allow any configuration with a sparse interconnection structure to be realized as part of a "universal" network. For example, universal networks of six elements with interconnection links to no more than three other elements can be exhibited which realize every possible six-element network in which elements are connected to no more than two other elements. Problem (1) is abstracted to determining what graphs may be embedded in a universal graph with $n$ nodes (elements) of degree (number of connections) $d$. Bounding relationships on $n$ and $d$ are derived. Problem (2) can be abstracted to trying to minimize the largest value of the number of PE's which one must pass through in communicating between two elements of a network. This problem can be considered in terms of the diameter of a graph. The problem...
then becomes the synthesis of graphs with prescribed diameters but a constrained degree of a node. Various synthesis (construction) techniques and bounds are described.

Problem (3) concerns permutation networks. Kautz et al. and Waksman utilized the basic permutation cell shown in Figure 8 to build interconnection arrays. The cell has two states defined as “cross” and “bend.” A number of different array types have been investigated, including triangular, diamond, rectangular, pruned rectangular, rhomboidal, square, almost square, Bose-Nelson, and rearrangeable arrays.

The triangular array is shown in Figure 9. The proof of its permutation capability is inductive. Clearly, the left most cell can perform the permutation for \( n = 2 \). Assume that \( n - 2 \) columns can perform any permutation of \( n - 1 \) elements, then by adding column \( n - 1 \) to the array the variable \( X_s \) may be switched into the sequence \( Y_1, \ldots, Y_{n-2} \) at any point, thereby adding an element and making the sequence \( X_1, \ldots, X_s \) map 1 to 1 and onto the sequence \( Y_1, \ldots, Y_s \) in any fashion (permutation). The number of cells required is \( \frac{1}{2}(n^2 - n) \). The other arrays studied were variations that altered the level of interconnectivity and the number of cells in the arrays. Further, the regularity of the structure was manipulated and studied. The rearrangeable array was essentially an implementation of Benes' rearrangeable networks described earlier.

**Thurber's programmable indexing networks**

Thurber introduced a network called an “indexing network” to overcome some deficiencies of \( n \) to \( n \) interconnection networks. Limitations of interconnection networks typically are that input words cannot be repeated or deleted at the output. Blanks cannot be inserted into the output and the number of input words and the number of output words must be equal. The indexing network differs from the permuter in that input words can be repeated or deleted and blanks can be inserted in the output. For an indexing net-
work, the number of input words \((n)\) has no special relation to the number of output words \((m)\). The non-blank output words may appear in many contiguous subsets of the output words (these subsets could be empty). Also, it is desirable to process data during its routing (e.g., matrix transposition in addition to simple sorting or merging).

Thurber proposed two general solutions. The first is essentially a circulating storage device with a map. The device contains an input storage device and an output storage device. The map determines how items are moved from the input to the output. Shift registers were envisioned as being used. The speed of the solution was dependent upon the amount of parallelism built into the network.

If \(N\) is a network with \(n\) inputs and \(m\) outputs then the output position map (OPM) is a vector containing \(m\) distinct cells with \(\log_2(n+1)\) binary bits per cell. Each cell contains the binary code corresponding to the input value desired in the corresponding output cell. \(\log_2(n+1)\) bits are needed since the \(n\) inputs and the 0 (blank) must have a code so that they can be specified as output values if desired.

Figure 10 is an example of a typical indexing network. The input word registers are shifted until the data tag matches the OPM value then the value is transferred and the OPM advanced. Obviously, this process is sequential and could require \(n \times m\) shift cycles. Variations on this theme include bidirectional shifting of the registers and increased hardware parallelism.

The second solution was called the “Splitter” and is shown in Figure 11. This solution required that each piece of data be furnished with a map (tag) called an input position map. The input position map (IPM) is a set of binary codes associated with the input data of a network that specifies the position (or positions) that the data is to be transferred to in the set of output registers. The allowable data transfers are indicated in Figure 12. The Splitter operates by transferring (at each stage) the top most piece of data according to the map. Eventually as indicated in Figure 11, the input sequence is reordered. Pipelining is possible through the network. The most difficult and limiting part of these solutions is that the maps must be preconstructed to sort the data based upon the tags.

Thurber's cascaded permutation network

Thurber studied the problem of constructing a permutation network from a cascade of identical cells such as shown in Figure 13. Figure 14 indicates the implementation of one
possible permutation and Figure 15 indicates the effect of the composition operator. Given the knowledge of how the basic permutation is realized and composed, the problem then resolves itself into selecting (via permutation group theory) a base which spans the space of all permutations, but which may be economically realized.

Three obvious solutions for the base were rejected. These were:

1. The set of $n!$ permutations would allow a single cell in the cascade but require many pins and large amounts of logic.
2. The set of two permutations
   \[
   \begin{pmatrix}
   1234 \ldots n \\
   2134 \ldots n
   \end{pmatrix}
   \quad \text{and} \quad
   \begin{pmatrix}
   1234 \ldots n-1 \ n \\
   2345 \ldots n \ 1
   \end{pmatrix}
   \]
   because this required cascades of length $n!$.
3. A solution based upon cardinality of permutation orbits because there is no generation algorithm for control.

Solutions selected were:

1. A base of $n$ elements in which a permutation of $n$ elements is inductively implemented using a permutation of $n-1$ elements and the permutation (3210...n-1)
2. A base of $2n-4$ elements in which permutations were added to the base of (1) above to reduce the cascade length.
3. A base chosen so that the cascade length did not exceed $n-1$ cells. This solution increased the base size to approximately $n^2$.

Detailed examples, calculations of length, control algorithms, etc., are given in Reference 18.

Tarjan's sorting network

Tarjan considered the problem of sorting a sequence of numbers using stacks and queues. Three types of basic elements were used. These were queues (FIFO elements), stacks (LIFO elements), and a switchyard. A switchyard is represented by a directed graph with a unique source and sink in which a vertex represents a siding. A siding can be either a queue or stack. Further assumptions include (1) the source and sink sidings must be queues and (2) storage may only occur at vertices of the graph. Tarjan concentrated on determining properties of sorting networks and their relationship to the ability to sort sequences. No specific networks are derived in the paper. The paper presents a good insight into the difficulty of describing general properties of sortable sequences. One important result derived is that if $m \geq 2$, then a sequence of length $3 \cdot 2^{m-1}$ (or less) may be sorted using $m$ stacks in series.

Harada's sequential permutation networks

Harada proposed two permutation networks. The first is a representation of the general permutation $(1,2,3,\ldots,r)$. The second is a lexicographic network representation. Its behavior may be computationally described as an explicit representation of the factorial counting function

\[
N = \sum_{h=1}^{n} A_h \cdot (h-1)! < n!,
\]

where $A_h$ is a $h$-nary factorial digit and $0 \leq A_h \leq (n-1)$.


Figure 14—Permutation cell

(\(X_1 X_2 X_3 X_4 X_5 X_6\) \(\sim\) \(123456\)
\(Y_1 Y_2 Y_3 Y_4 Y_5 Y_6\) \(\sim\) \(562413\))

Figure 15—Composition operator example

(\(123456\) \(562413\) \(\sim\) \(123456\)
\(654123\) \(\sim\) \(123456\)
\(235164\))
Harada provides a detailed examination of the relation­ship between the two networks and their control states. His analysis begins with the observation that given the ability to generate all permutations of \(k-1\) elements, another state in the network can be constructed (utilizing more elements) which can construct all permutations of \(k\) elements. The networks employed are two versions of the Kautz\(^4\) et al. triangular array. As shown in Figure 8, each cell may bend or pass a variable. Two complementary networks (shown in Figures 16 and 17) are defined. The network of Figure 16 works like the triangular array of Kautz. The network of Figure 17 is the lexicographic network.

The goal of the study was to define a network which would produce permutations for computational purposes not an interconnection network, but the network could be used as an interconnection network. The following conclusions were reached about each network:

1. each could generate all \(n!\) permutations
2. a permutation could be derived from its predecessor in a single clock step.
3. the permutor of Figure 16 represents the equation \((1, 2, 3, \ldots, n)\) and is input oriented.
4. the permutor of Figure 17 represents the factorial counting function and is output oriented.

Bandyopadhyay's permutor network

Bandyopadhyay\(^8\) et al. discussed the implementation of the network shown in Figure 18. The concept was based upon the generation of a permutation of \(K\) elements by being able to generate a permutation of \(K-1\) elements. The selector cells were realized using cellular logic arrays. The advantage of this approach is the small number of control lines. The disadvantage in comparison to Thurber's\(^7\) cascade approach is that all cells are unique and thus the concept is suitable for LSI implementation only if cellular arrays can be used in the implementation as proposed\(^9\).
Smith's sorting network analysis

Smith\(^2\) considers a number of problems concerning the concept of a sorting network. Whereas, most of the other references are mainly interested in how to synthesize sorting networks, Smith was interested in whether a network sorts or not. One of the most important results of his investigation is that no comparator network containing one or more comparators can sort without sorting some other assignment. More importantly, the paper notes some of the reasons why the problem of designing sorting networks is so difficult: i.e.,

1. lack of algebraic structure on the subject;
2. lack of tools to manipulate composite networks of comparators; and
3. lack of tools to manipulate sets of network assignments.

Smith did, however, discover that it seemed easier to approach the subject from the standpoint of Boolean expressions rather than from the standpoint of partially ordered sets. Furthermore, he did not see any promising opportunities for resolution of even the simplest questions, such as, how many comparators are required to construct a sorting network.

Lawrie's \(\Omega\) networks

Lawrie\(^3\) developed a set of connection networks and algorithms for their synthesis based upon the concept of the mathematical notion of a "\(\Omega\) base" representation of integers, the research was being done to determine effective means of interconnecting processing elements in highly parallel processor configurations.

Lawrie shows that if \(R_n\) is an ordered set of integer factors of \(n (R_n = (p_1, \ldots , p_k)\) such that \(p_1 \cdot p_2 \cdot \ldots \cdot p_k = n\) then \(\Omega(R_n)\) may be defined as the set \(\Omega(R_n) = (W_1/W_k = 1, W_1/W_{k+1} = 1, \ldots , W_1/W_{n+1} = 1)\) where \(0 < i \leq k - 1\) and \(W_i = \prod_{j=i+1}^{k} P_j\). Also note that \(W_k = n\) and \(W_l/W_i = 1\) means that there exists an integer, \(a\), such that \(W_l \cdot a = W_k\). Using \(\Omega(R_n)\), Lawrie develops a connection network algorithm. The algorithm constructs a \(k\) stage network (stages numbered 1, 2, \ldots , \(k\) from left to right). The \(i\)th stage is composed of \(n/p_i\) crossbar switches, each switch \(p_i\) by \(p_i\). At each stage the \(n/p_i\) crossbar switch networks have their inputs and outputs labeled from 0 to \(n-1\) (e.g., the inputs and outputs of switch number 2 are \(p_0, p_0 + 1, \ldots , 2p_i - 1\)). Connections are made by connecting output \(j\) of stage \(i\) to input \(l\) of stage \(i + 1\) where \(l = (j + W_{i+1}) \cdot W_{i+1} + (j \mod p_i) \cdot W_i + (j \mod W_{i+1}) + p_i\). Where \(X \mod Y\) is taken to mean the integer part of the quotient. A special set of connections must be computed for stage 1. A control algorithm is developed by constructing a \(n\)-set, which represents a mapping or connection between \(n\) input and \(n\) output nodes. \(n\)-sets can be used to describe the control of the network and to detect conflicts in routing. A detailed study of the properties, effectiveness, and construction of \(\Omega\) networks is given in Reference 23.

One \(\Omega\) network which Lawrie implemented is shown in Figure 19. This is a bit serial network constructed from building blocks that transmit and/or block signals in response to the strobe or reset at each cell. Transmissions allowed are \(A = C, B = D,\) or \(A = D, B = C\). Conflicts can occur but may also be resolved in this network scheme.

\[
\begin{align*}
0 &\leftrightarrow A
1 &\leftrightarrow B
2 &\leftrightarrow C
3 &\leftrightarrow D
4 &\leftrightarrow 0
5 &\leftrightarrow 1
6 &\leftrightarrow 2
7 &\leftrightarrow 3
8 &\leftrightarrow 4
9 &\leftrightarrow 5
10 &\leftrightarrow 6
11 &\leftrightarrow 7
\end{align*}
\]

Figure 19- \(\Omega\) network
There are a number of things to note about the $\Omega$ network of Figure 19. First, topologically, its last stages represent bitonic sorters and rearrangeable networks. Further when it is redrawn as shown in Figure 20, it contains the perfect shuffle interconnections and the interconnections become quite regular.

There are many forms of $\Omega$ networks. Lawrie has analyzed numerous variations which minimize various pertinent factors. His work is probably the best analytic study of interconnection networks performed to date. It provides us with the most advanced mathematical tools so far developed in the area of sorting and interconnection networks. Crossbar switches, bitonic sorters, etc., can be viewed as special cases of $\Omega$ networks.

A "banyan" as defined by Webster is an East Indian fig tree whose branches grow shoots that take root and become new trunks. A banyan can be illustrated graphically. A graph of a banyan is a Hasse diagram of a partial ordering in which there is one and only one path from any base to any apex. A base constitutes any vertex with no incident arcs. An apex is any vertex without arcs incident from it. All other vertices are called intermediates. Figure 21 is a banyan. The significance of banyan networks is that their tree shaped structure provides the possibility of allowing low propagation delays with limited fan out systems. Further, priority hardware can be included to minimize conflicts and conflict detection schemes can be devised.

Large banyan networks may be synthesized recursively from smaller networks. Goke and Lipovski prove a number of theorems relating to banyan networks, which will allow synthesis of large banyan networks and determination of the networks potential for conflicts.

L-level banyan (a banyan whose vertices are arranged in levels so that arcs can only exist between vertices in adjacent levels) are considered for propagation delay factors (distance between base and apex is the critical factor), conflicts (a criteria to avoid conflicts and enhance performance is derived), and speed factors.

There are two quite important banyan networks. These are the SW and CC (cylindrical crosshatch) structures. The SW structure recursively expands to a crossbar switch as illustrated in Figure 22. The CC structure is rectangular and contains $N = S^2$ vertices in each level. If $V_{k0}, V_{k1}, \ldots, V_{kn-1}$ are the vertices in each level $K$ of an $L$-level CC banyan structure then there is an arc from a vertex $V_{ki}$ to a vertex $V_{k+i+mS^2}$ in the level above whenever $j = i + mS^2 \pmod{n}$ for some $m = 0, 1, \ldots, S - 1$. Furthermore, crossbar switches, bitonic sorters, etc., are able to be described and analyzed as banyans.
Table I—Comparison of Interconnection Network Approaches

<table>
<thead>
<tr>
<th>Network Type</th>
<th>Cells Per Connection</th>
<th>Cells Per Cell</th>
<th>Number of Cells</th>
<th>Upper Bound on Cycles</th>
<th>Lower Bound on Cycles</th>
<th>Common Element</th>
<th>Common Element in Some Computer Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barrel Shifter</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
</tr>
<tr>
<td>Crossbar</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
</tr>
<tr>
<td>Benes</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
</tr>
<tr>
<td>Batcher</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
</tr>
<tr>
<td>Kautz</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
</tr>
<tr>
<td>Barbell</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
</tr>
<tr>
<td>Thacker</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
</tr>
<tr>
<td>Lambda</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
</tr>
<tr>
<td>Bob</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
<td>10^3 + n * x</td>
</tr>
</tbody>
</table>

APPLICATIONS

There are a number of applications for which sorting and interconnection networks may be used. Some typical ones are listed below:

1. Sorting data
2. Switching networks with priority (conflict) resolution
3. Multicasting memories
4. Content addressable memories
5. Recirculating memories
6. Permutation generators
7. Restructurable bussing structures
8. Interconnection networks for parallel processors

Some actual problems requiring (or solved by) application of interconnection networks may be found in References 25–32. Further references of interest pertinent to the subject of interconnection networks begin with Reference 33 and continue to Reference 48.

COMPARISON OF INTER_CONNECTION NETWORKS

It is very difficult to make a comparison of interconnection networks for the following reasons:

1. There is very little theory and analytic tools available.
2. The networks were mainly developed ad hoc and thus satisfy varying requirements, making them even more difficult to compare due to their different capabilities.
3. We have no baseline networks to reference.
4. There are really two structures to consider (control and data routing).

Table I has been constructed to give the reader a feel for some global differences in the networks. Presented in this table are some very gross comparisons of the "order-of-magnitude" type. The table is admittedly quite rough, because the detailed comparison of such networks is far beyond the scope of this paper. Barrel shifters and crossbar switches have been included to provide a baseline feel for the networks complexity. Notably, the order-of-magnitude properties of the networks are quite similar so that differences in the networks lie in the areas of ease of control, detailed capabilities, practicality, ease of reduction to design, required number of gates, ability to utilize off-the-shelf parts for construction, number of pins, size, weight, power, etc. Furthermore, data transmission time and the upper bound on number of cycles are identical, except one must note that you need to multiply data transmission time by a time factor T for each stage to account for time delays and to obtain a realistic data transmission time figure. T may be different for each cell type thus producing different speed networks.

CONCLUSION

The most important issue clouding the future of interconnection networks is their practicality. Numerous ad hoc approaches have been developed which demonstrate that interconnection networks can be designed but that reduction of such networks to economically viable units is very difficult. A number of theoretical developments have been introduced which provide good methodologies for analytically designing interconnection networks. Notably, the complexity, delay time, number of units required, etc. do not differ significantly between approaches. Thus it is time to consider the practical problems associated with interconnection networks. Some of the practical issues that must be resolved before interconnection networks can become viable components of a computer system include:

1. Determination whether any additional functions are appropriate to be included in the networks
2. Design of basic building block modules which provide good gate to pin ratios, are low cost, and provide the ability to be utilized to construct larger networks.
3. Development of cost effective LSI partitioning of the networks and their basic building blocks
4. Development of simple, easily changeable control algorithms
5. Investigation of the problems involved with actually transmitting signals through the networks.

To date work on interconnection networks has generally concentrated on achieving the network functionality and ignored the problems associated with implementing the networks. Economically viable networks have yet to be achieved. Furthermore, many difficult problems remain to be solved before the networks approach practicality. From the view point of the network's functional characteristics, real progress is being made. Some of the recent advances include a number of early network designs as special cases. Thus a strong analytic base has been laid.

In distributed systems, systems of many small processors, or systems constructed from modular logic, the interconnection logic may be at least as expensive as the logic in the processors themselves, thus the recurring hardware cost of
modular computer networks may be two to ten times that of a general purpose sequential machine capable of doing the same job. Therefore, deriving economical interconnection networks and bus structures will be more important than ever due to advances in LSI technology. The theoretic future of interconnection networks appears bright, but the real challenge lies in reducing to practice economical interconnection networks which will make possible digital systems with a high degree of interconnectivity.

REFERENCES


