Processor-Memory Interconnections for Multiprocessors

Janak H. Patel
School of Electrical Engineering
West Lafayette, IN 47907

Abstract -- A new class of interconnection networks is proposed for processor to memory communication in multiprocessing systems. These networks allow a direct link between any processor to any memory module. The cost of these networks is considerably less than that of full crossbars. Moreover, the design and control of these networks is simple. The proposed networks and the full crossbars are analyzed with respect to the bandwidth and the cost.

I. Introduction

With the advent of low cost microprocessors, the architectures involving multiple processors are becoming very attractive. Several organizations have been implemented or proposed. Principally among these are parallel (SIMD) type processors, computer networking and multiprocessor organizations. In this paper we focus our attention to multiprocessors.

The principle characteristics of a multiprocessor system is the ability of each processor to share a single main memory. This sharing capability is provided through a interconnection network between the processor and the memory modules, which logically looks like Figure 1. The function of the switch is to provide a logical link between any processor and any memory module. There are several different physical forms available for the processor-memory switch; the least expensive of which is the time-shared bus. However, a time-shared bus has a very limited transfer rate, which is inadequate for even small number of processors. At the other end of the bandwidth spectrum is the full crossbar switch, which is also the most expensive switch. In fact, considering the current low costs of microprocessors and memories, a crossbar would probably cost more than the rest of the system components combined. Therefore it is very difficult to justify the use of a crossbar for large multiprocessing systems. It is the absence of a switch with reasonable cost and performance, which has prevented the growth of large multimicroprocessor systems. To circumvent the high cost of switch, some "loosely coupled" systems have been proposed. In these systems, sharing of main memory is somewhat restricted, for example some memory accesses may be fast and direct while many other references may be slow, indirect and may even involve operating system intervention. There is considerable research on the permutation networks for parallel (SIMD) processors but almost no research on processor-memory interconnections requiring random access capabilities.

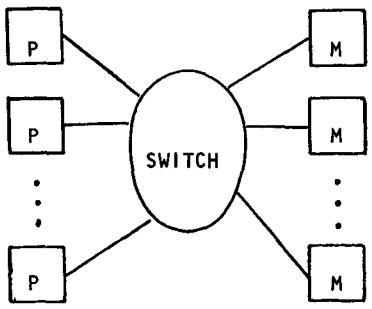

Figure 1. Logical organization of a multiprocessor.

In this paper we propose a class of interconnection networks, termed delta networks which are far less expensive then full crossbars and are modular and easy to control. These networks are analyzed to determine the effective bandwidth and are compared with the effective bandwidth of full crossbars. We also describe the implementation details of delta networks and their cost-effectiveness.

II. Principle of Operation

Before we define delta networks, let us study the basic principle involved in the construction and control of delta networks. Consider a 2 x 2 crossbar switch (Fig. 2). This 2 x 2 switch has the capability of connecting the input A to either the output labeled 0 or output labeled 1, depending on the value of some control bit of the input A. If the control bit is 0 then the input is connected to the upper output and if 1 then it is connected to the lower output. The same description applies to terminal B, but for the time being ignore the existence of B. It is straightforward to construct a 1-by-2^n demultiplexer using the above described 2 x 2 module. This is done by making a binary tree of this module. For example Figure 3 shows a 1 x 8 demultiplexer tree. The destinations are marked in binary. If the source A requires to connect to destination $(d_2d_1d_0)_2$ then the root node is controlled by bit d_2, the second stage modules are controlled by bit d_1 and the last stage modules are controlled by bit d_0. It is clear that, A can be connected to any one of the eight output terminals. It is also obvious that the lower input terminal of the root-node also can be switched to any one of the 8 outputs.

CH1394-6/79/0000-0168 $00.75©1979 IEEE

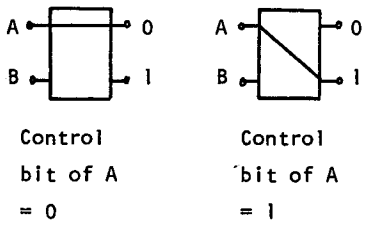

Figure 2. A 2 x 2 crossbar.

At this point we add another capability to the basic 2 x 2 module, the capability to arbitrate between conflicting requests. If both inputs require the same output terminal, then only one of them will be connected and the other will be blocked or rejected. The probability of blocking and the logic for arbitration is treated later on.

Now consider constructing an 8 x 8 network using 2 x 2 switches, the principle used is the same as that of Figure 3. Every additional input must also have its own demultiplexer tree to connect to any one of the eight outputs. Basically the construction works as follows. Start with a demultiplexer tree, then for each additional input superimpose a demultiplexer tree on the partially constructed network. One may use the already existing links as part of the new tree or add extra links and modules if needed. We have redrawn the tree of Figure 3 as Figure 4a. The addition of next tree is shown with heavy lines in Figure 4b. This procedure is continued until the final 8 x 8 network of Figure 4d results. The only restriction which must be strictly followed during this construction is that, if a 2 x 2 module has its inputs coming from other modules then both inputs must come from upper terminals of other modules or both must be lower terminals of other modules. (All upper output terminals are understood to have label 0 and the lower terminals, label 1.) Other than this there is considerable freedom in establishing links between the stages of the network. In the construction of the above 8 x 8 network we had the benefit of some hindsight that 12 modules are necessary and sufficient to build this network. If more modules are used then some inputs of some modules will remain unconnected. We could have stopped in the middle of the construction to obtain a 4 x 8 or a 6 x 8 network, such as Figure 4b and 4c, however in each case some inputs of the 2 x 2 modules will remain unutilized.

We term, the networks constructed in the above manner digit controlled or simply delta networks, since each module is controlled by a single digit from the destination address. Furthermore, no external or global control is required. Digit controlled networks are not new; Lawrie's omega networks [1] and Pease's indirect binary n-cube [2] are subsets of delta networks.

Note that, network of Figure 4d does not allow an identity permutation, that is, the connection 0 to 0, 1 to 1, ..., 7 to 7 at the same time. An identity permutation is useful if say memory module 0 is a "favorite" module of processor 0, and module 1 that of processor 1 and so on. Thus identity permutation allows most of the memory references to be made without conflict. A simple renaming of the inputs of Figure 4d will allow an identity permutation. This is shown in Figure 5; in here if all 2 x 2 switches were in the straight (=) position then an identity permutation is generated. As a matter of fact, since one and only one path is available from any source to any destination, every different setting of the form X and = generates a different permutation. Thus the network of Figure 5 generates 2^{12} distinct permutations. This brings us to another somewhat unrelated topic of permutation networks. The procedure to construct a delta network can be used to generate different permutation networks. For example we could have started with a tree in which the first stage was controlled by bit d_1 the second by d_0 and third by d_2. This would of course require relabeling the outputs. But does this really produce a "different" network? Siegel [3] has shown that by a simple address transformation the newtorks of Lawrie [1] and that of Pease [2] can be made equivalent, i.e., they produce the same set of permutations. As far as we know, the network of Figure 5 cannot be made equivalent to either Lawrie's or Pease's network by a simple address transformation. It is quite possible, that there are only two non-equivalent 8 x 8 delta networks, namely Lawrie's omega network and the network of Figure 5. We shall not pursue this subject any further, as our primary interest lies in the random access capabilities of these networks and not permutations.

III. Design and Description of Delta Networks

So far we have not defined the delta networks in a formal and rigorous manner. For the purposes of this paper we define them as follows.

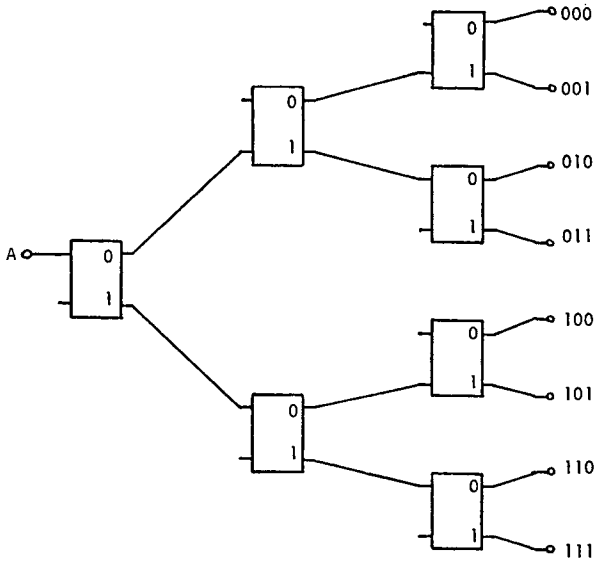

Figure 3. 1-by-8 demultiplexer.

Let a b x b crossbar module have the capability to connect any input to any one of the b outputs. Let the outputs be labeled 0, 1, ..., b-1. An input is connected to the output labeled d if the control digit supplied by the input is d, where d is a base-b digit. Moreover, a b x b module also arbitrates between conflicting requests by accepting some and rejecting others.

A delta network is a b^n x b^n network made up of nb^{n-1}, b x b crossbar switches, arranged in n stages, b^{n-1} switches to a stage. The link pattern between stages is such that any source can be connected to any destination, where each b x b modules connects an input to one of its b outputs depending on a single base-b digit taken from the destination address.

The construction of a b^n x b^n delta network follows the principle presented in the previous section. Informally, the procedure can be described as follows.

Construct a b-ary demultiplexer tree using b x b crossbar switches. A b-ary tree has b branches for every node. For a 1-by-b^n demultiplexer, the tree has n levels. Each level is controlled by a distinct base-b digit taken from the destination address. For every additional input source, superimpose a new tree on the partially completed network. Each superimposition must satisfy two conditions. 1. No more than b^{n-1} modules may be used at any level and no more than n levels are created. 2. Each b x b module which receives inputs from other b x b modules, must have all its inputs connected

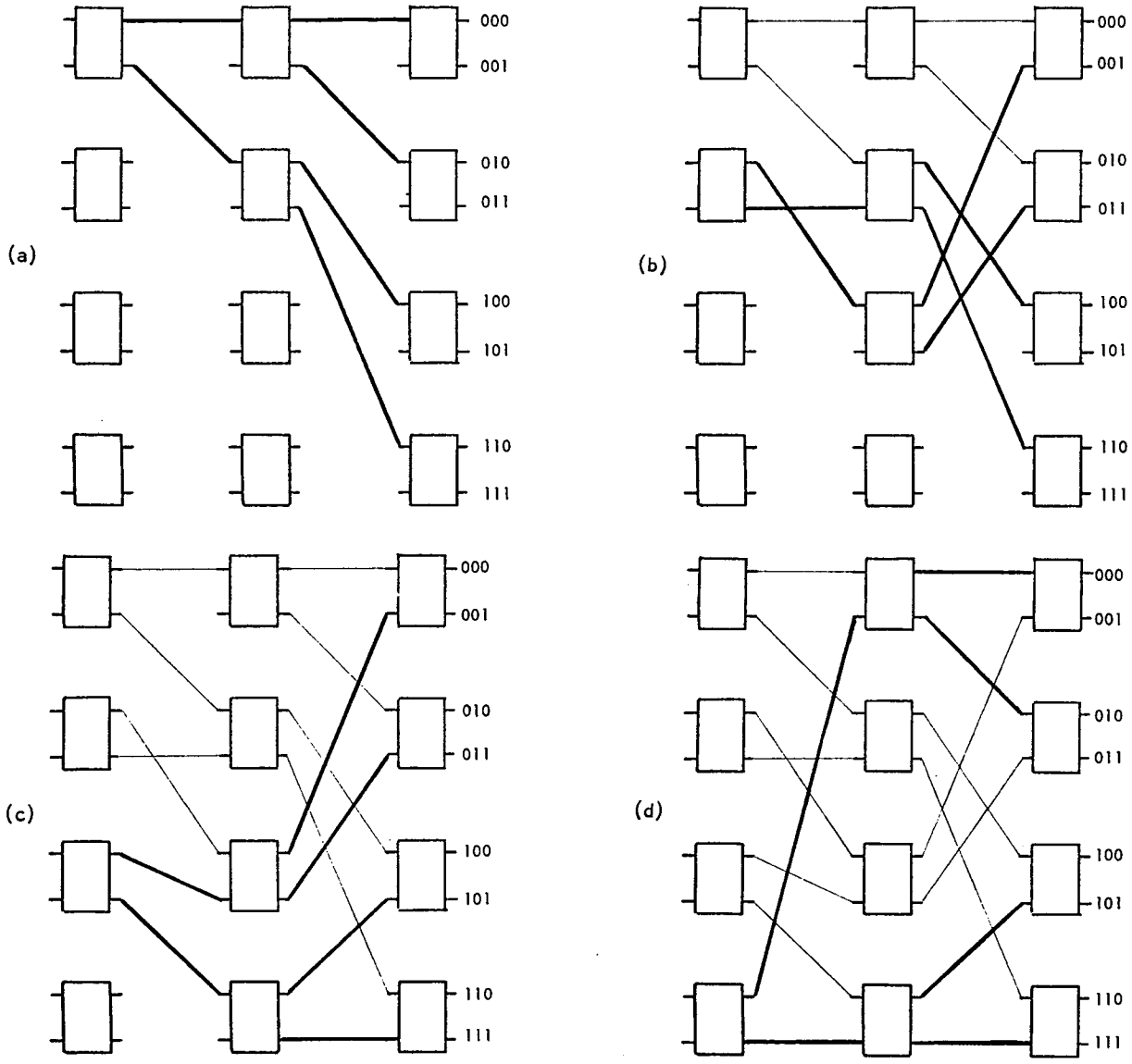

Figure 4. Construction of an 8 x 8 delta network.

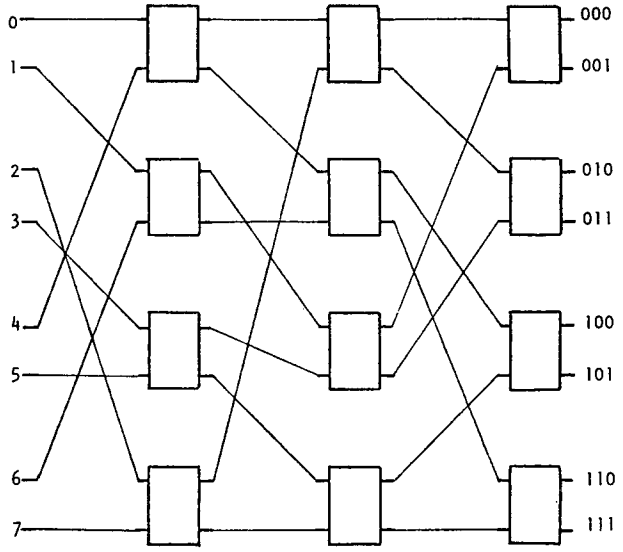

Figure 5. An 8 x 8 delta network to allow identity permutation.

to identically labeled outputs, where the outputs of each b x b module are labeled 0, 1, ..., b-1, as was described earlier.

As one can see from the above construction procedure, there is a large number of link pattern available for a b^n x b^n delta network. It is natural to wonder if one topology is better than the others. We shall see later that, as far as probability of acceptance or blocking for random access is concerned, all delta networks are identical. However, different topologies may have different permutation capabilities. An important characteristic of delta networks which results from a property of trees used in the construction is as follows.

In a delta network, there is one and only one path from a source to a destination.

As a result of the above characteristic, every different setting of a b x b switch results in a different permutation. Thus a b^n x b^n delta network generates $(b!)^{nb^{n-1}}$ distinct permutations. This number is a very small fraction of the all possible permutations of b^n inputs, even for small values of b and n. For example, the probability that a random permutation of 32 inputs can be generated by a 2^5 x 2^5 delta network is 4.6×10^{-12}. From this, it would indeed be erroneous to make any conclusion about the effectiveness of delta networks for random memory accesses.

The uniqueness of a path between a source and a destination simplifies the control and analysis of delta networks. However, the uniqueness happens to be a weak point from the reliability standpoint. The reliability aspects of delta networks will be reported at a later date.

Since, the link pattern between stages of a delta network is of no particular concern to us, we may ask if there is some regular link pattern, which can be used between all stages and thus avoid the cumbersome construction procedure for every different delta network. There is indeed such a pattern which we describe below.

Let a q*r shuffle, denoted S_{q*r}, where q and r are some positive integers, be a permutation of qr indices <0, 1, 2, ..., (qr-1)>, defined as

$$S_{q*r}(i) = (qi + \left\lfloor \frac{i}{r} \right\rfloor) \bmod qr \qquad 0 \leq i \leq qr-1$$

where $s_{q*r}(i)$ is the position of i after the shuffle. A q*r shuffle can be viewed as a shuffle of qr cards in the following way. Divide the deck of qr cards in to q piles of r cards each; top r cards in the first pile, next r cards in the second pile and so on. Now pick the cards, one at a time from the top of each pile; the first card from top of pile one, second card from the top of pile two and so on in a circular fashion, until all cards are picked up. This new order of cards represents a S_{q*r} permutation of the previous order.

From the above description it is clear that S_{2*r} is the well known perfect shuffle (e.g. [4]). Not so obvious is the fact that S_{r*q} is an inverse permutation of S_{q*r}. That is,

$$S_{q*r}(S_{r*q}(i)) = i \qquad 0 \leq i \leq qr-1$$

The q*r shuffles, although never defined formally as such, are widely used in interconnection networks, for example between the stages of a Clos network [5]. We shall state without proof that a b^n x b^n delta network can be constructed by using a $b*b^{n-1}$ shuffle as the link pattern between every two stages. To use the function $S_{b*b^{n-1}}$, all the inputs and outputs of every stage must be labeled from top to bottom as $<0, 1, 2, \ldots, (b^n-1)>$. Figure 6 shows a general b^n x b^n delta network. Note that shuffle networks between stages are passive and not active like the stages themselves. Two delta networks, one 3^2 x 3^2 and one 2^3 x 2^3, derived from Figure 6 are shown in Figures 7 and 8; the interstage link pattern are respectively 3*3 shuffle and 2*4 shuffle. In Figure 6, by using a $b*b^{n-1}$ shuffle of the inputs just before the first stage, one can obtain the identity permutation. If such a transformation is used on the inputs of Figure 8 then it becomes topologically identical to Lawrie's 8 x 8 omega network.

IV. Implementation of Delta Networks

Within the current technological limitations, it is uneconomical to encode base b digits where b is not a power of 2. Thus in practice, a b x b crossbar module for a delta network is more costeffective if b is a power of 2, since our modules require base b digits for control. Again, due to the

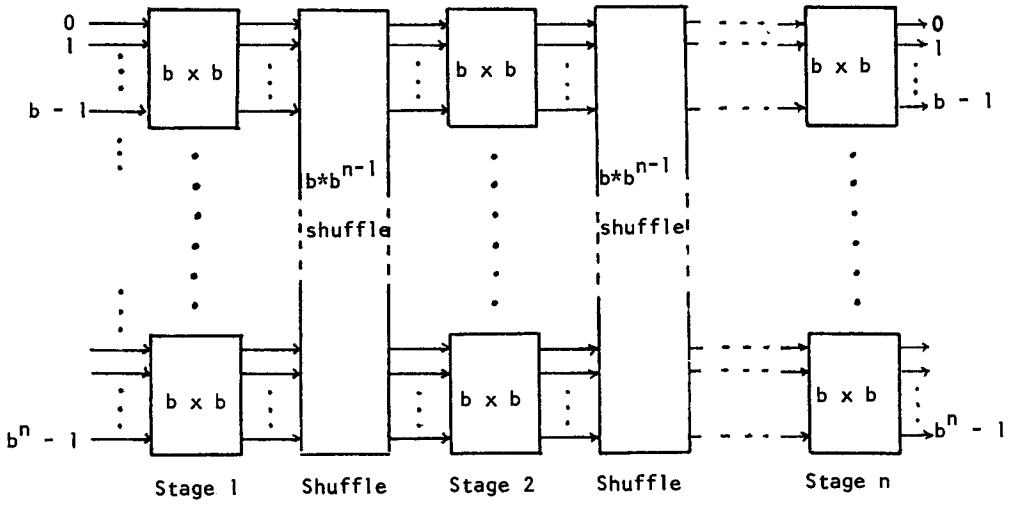

Figure 6. A b^n x b^n delta network.

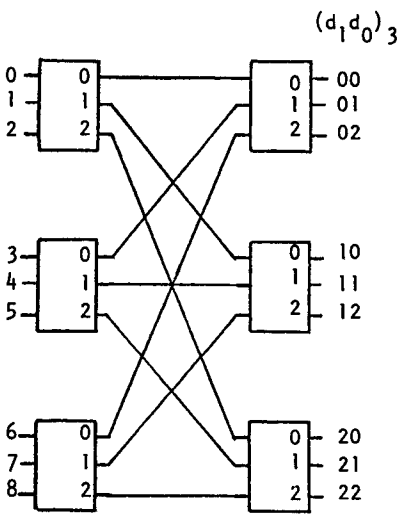

Figure 7. A 3^2 x 3^2 delta network.

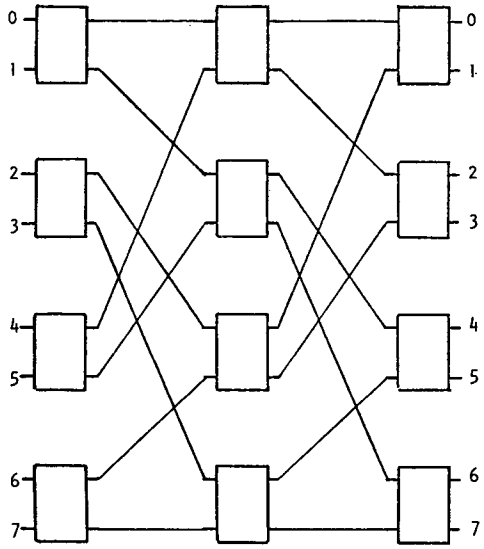

Figure 8. A 2^3 x 2^3 delta network.

cost and technological limitations, modules of size 8 x 8 or greater are not very practical at this time. This leaves 2 x 2 and 4 x 4 modules as the most likely candidates for implementation of delta networks. Here we give the functional and logical description of 2 x 2 modules. This in turn will be used to estimate the cost and delay factors of delta networks.

The functional block diagram of a 2 x 2 crossbar module of a delta network appears in Figure 9. All single lines in the figure are one bit lines. The double lines on INFO box, represent address lines, incoming and outgoing data lines, and a read/write control line. The data lines may or may not be bi-directional. The function of the INFO box is that of a simple 2 x 2 crossbar; if the input X is 1 then a cross connection exists and if X is 0 then a straight connection exists.

The function of the CONTROL box is to generate the signal X and provide arbitration. A request exists at an input port if the corresponding request line is 1. The destination digit provides the nature of the request; a 0 for the connection to upper output port and a 1 for the lower port. In case of conflict, the request r_0 is given the priority and a busy signal $b_1 = 1$ is supplied to the lower input port. A busy signal is eventually transmitted to the source which originated the blocked request. The logic equations for all the

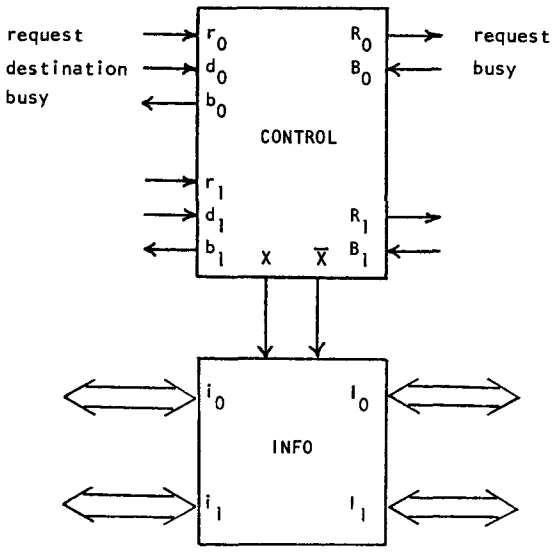

Figure 9. Details of 2 x 2 modules for delta networks.

labeled signals are given with the block diagram. For INFO box, the equations are given for left to right direction. The parallel generation of X and $\overline{X}$ reduces one gate level. Signal X and $\overline{X}$ are valid after 3 gate delays. Assuming that one level of buffer gates for X and $\overline{X}$ in the INFO box exists due to fanout limitations of X and $\overline{X}$, the total delay to establish the connections of INFO box is 6 gate delays, of which 4 gate delays are due to X and $\overline{X}$. Thus after the initial set up time, the data transfer requires only 2 gate delays per stage of the network.

The operation of a 2^n x 2^n delta networks using the above described 2 x 2 modules is as follows; recall that there are n stages in this network.

All processors requiring memory access must submit their requests at the same time by placing a 1 on the respective request lines. After 8n gate delays the busy signals are valid. If the busy line is 1, then the processor must resubmit its request. This can be accomplished simply by doing nothing, i.e., continue to hold the request line high. The read data is valid after 8n gate delays plus the memory access time if the busy signal is 0. Thus the operation of the implementation described here is synchronous, that is, the requests are issued at fixed intervals at the same time. An asynchronous implementation is preferable if the network has many stages. However, such an implementation would require storage buffers for addresses, data and control in every module and also a complex control module. Thus, the cost of such an implementation might well be excessive. We have analyzed only the synchronous networks in this paper.

V. Analysis of Delta Networks

In this section we analyze b^n x b^n delta networks for evaluating the probability of acceptance of a request and the expected bandwidth. The analysis is based on the following assumptions.

1. Each processor generates random and independent requests; the requests are uniformly distributed over all memory modules.

2. The mean request generation rate of each processor is m requests per cycle, where a cycle is the time required to pass through the network plus the time to access the memory plus the time to return to the source irrespective of whether a request is a read or a write. m is less than or equal to 1.

3. New requests are generated every cycle and submitted at the same time. At most 1 request can be generated by a processor during one cycle.

4. The requests which are blocked (not accepted) are ignored. That is, the requests issued at the next cycle are independent of the requests blocked.

The cycle time of assumption 2 can be evaluated from the implementation details of the particular network. For example the cycle time of a 2^n x 2^n delta network is 8n gate delays plus memory access time. The assumptions 2 and 3 together imply that the mean request generation rate m is the probability that a request is generated by a processor during a cycle. The fourth assumption is there to simplify the analysis. In practice of course the rejected requests must be resubmitted during next cycle; thus the independent request assumption will not hold. However, to assume otherwise, would make the analysis if not impossible, certainly very difficult. Moreover, simulation studies done by us and by others [6] for similar problems have shown that the probability of acceptance is only slightly lowered if the fourth assumption above is omitted. Thus the results of the analysis are fairly reliable and they provide a good measure for comparing different networks.

First we analyze the 2^n x 2^n delta networks in some details and then we shall present the generalized case.

Let P_A be the probability that a request will be accepted. The bandwidth BW of a N x N network, defined as the expected number of requests accepted per cycle is then mNP_A, where m is the mean rate of request generation of each processor.

Let $p(0|i)$ be the probability that given i requests at a 2 x 2 module M, nothing is sent on a particular output line of M. Since both output lines of M are equally likely to be requested, we have,

$$p(0|0) = 1 \qquad p(0|1) = \frac{1}{2} \qquad p(0|2) = \frac{1}{4} \quad (1)$$

Let $p(1|i)$ be the probability that given i requests at module M, a request is sent out on a particular output line of M. Clearly,

$$p(1|i) = 1 - p(0|i). \quad (1)$$

Moreover, E(i), the expected number of requests accepted by module M, given i requests at M is:

$$2p(1|i) \quad (2)$$

Now let $q_h(k)$ be the probability that k requests arrive at a module of stage h and let $P_a(h)$ be the probability that a request arriving at stage h is accepted by stage h. Then,

$$P_a(h) = \frac{\left(\begin{array}{c}\text{expected number of requests}\\ \text{accepted by a module of stage h}\end{array}\right)}{\left(\begin{array}{c}\text{expected number of requests}\\ \text{arriving at a module of stage h}\end{array}\right)}$$

From (2) and the definition of $q_h(k)$ we have

$$P_a(h) = \frac{E(1)\cdot q_h(1) + E(2)\cdot q_h(2)}{1\cdot q_h(1) + 2\cdot q_h(2)} \quad (3)$$

The probability $q_h(k)$ can be evaluated recursively by the following procedure.

Let $r(k|i,j)$ be the probability, given i requests at module M_1 and j requests at module M_2 of a stage, that k requests reach module A of the next stage (figure 10a). Then,

$$\begin{aligned} r(0|i,j) &= p(0|i)\cdot p(0|j) \\ r(1|i,j) &= p(0|i)\cdot p(1|j) + p(1|i)\cdot p(0|j) \quad (4) \\ r(2|i,j) &= p(1|i)\cdot p(1|j) \end{aligned}$$

Now q_{h+1} can be expressed in terms of q_h as follows.

$$q_{h+1}(k) = \sum_{0\le i,j\le 2} r(k|i,j)\cdot q_h(i)\cdot q_h(j) \quad (5)$$

The initial conditions of q's is derived from the mean request generation rate m of each processor. Since, m is the probability that a request is generated by a processor, the distribution of requests arriving at a 2 x 2 module of the first stage can be expressed as:

$$q_1(0) = (1-m)^2 \qquad q_1(1) = 2m(1-m) \qquad q_1(2) = m^2 \quad (6)$$

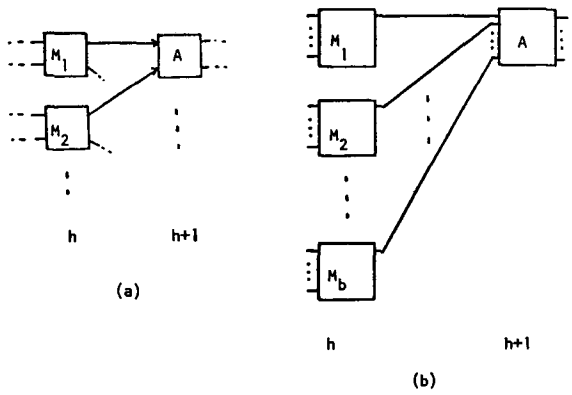

Figure 10. Analysis of delta networks.

Finally, the overall probability of acceptance P_A, of 2^n x 2^n delta network is given by the following product.

$$P_A = \prod_{1\le h\le n} P_a(h) \quad (7)$$

Letting $N = 2^n$, bandwidth BW, the expected number of requests accepted per cycle is as follows.

$$BW = mNP_A \quad (8)$$

Equations (1) thru (8) above are sufficient to compute the probability of acceptance of a request and the expected bandwidth of any 2^n x 2^n delta network, given mean request generation rate m.

The above procedure can be generalized, so that it is applicable to any b^n x b^n delta network. We give it here in a concise fashion. The best way to understand each equation is to first understand the corresponding equation for 2^n x 2^n case. Each equation number below is suffixed by b and exactly corresponds to the same equation number without suffix of 2^n x 2^n case. An important characteristic of any b^n x b^n delta network is that an arbitrary module A of any stage (except the first stage) has the connection pattern of Figure 10b, that is, no two inputs to A come from the same module. This fact is essential for the following analysis.

$p(0|i)$: the probability, given i requests at a module, that a particular output line of that module is not requested.

$p(1|i)$: the probability, given i requests at a module, that a particular output line of that module is requested.

E(i): the expected number of requests accepted by a module; given i requests at that module.

$q_h(k)$: the probability that k requests arrive during a cycle at a module of stage h.

$r(k|i_1,\ldots,i_b)$: the probability, given i_1 requests at M_1, i_2 requests at M_2, ..., i_b requests at M_b,

that k requests arrive at module A of next stage (see Figure 10b).
$P_a(h)$: the probability that a request arriving at stage h is accepted by stage h.

Since only one request can be present on any line, it is clear that at most b requests can arrive at any module in one cycle. Thus $0 \le i,k,i_1, \ldots, i_b \le b$ in all of the definitions above and equations below.

$$p(0|i) = (1/b)^i \qquad p(1|i) = 1 - (1/b)^i \tag{1b}$$

$$E(i) = b \cdot p(1|i) \tag{2b}$$

$$P_a(h) = \frac{\sum_{0\le i\le b} E(i)\cdot q_h(i)}{\sum_{0\le i\le b} i\cdot q_h(i)} \tag{3b}$$

$$r(k|i_1,\ldots,i_b) = \sum_{\substack{0\le j_1,\ldots,j_b\le b \\ (j_1+\ldots+j_b)=k}} p(j_1|i_1)\cdot p(j_2|i_2)\cdots p(j_b|i_b) \tag{4b}$$

$$q_{h+1}(k) = \sum_{0\le i_1,\ldots,i_b\le b} r(k|i_1,\ldots,i_b)\cdot q_h(i_1)\cdot q_h(i_2)\cdots q_h(i_b) \tag{5b}$$

Initial conditions: $q_1(i) = \binom{b}{i} m^i (1-m)^{b-i}$ (6b)

Probability of Acceptance: $P_A = \prod_{1\le h\le n} P_a(h)$ (7b)

Expected Bandwidth:

$$BW = mb^n P_A \text{ requests per cycle} \tag{8b}$$

In the above, $\binom{b}{i}$ is the binomial coefficient and m is the mean request generation rate of each processor.

VI. Analysis of Full Crossbar Networks

For the purpose of comparison we analyze here the full crossbar networks of size N x N under assumptions identical to those of delta networks, stated in the beginning of the previous section. Recall that m is the mean request generation rate of each processor, that is, the probability that a processor generates a request during a cycle is m. Let q(i) be the probability that i requests arrive during one cycle. Then,

$$q(i) = \binom{N}{i} m^i (1-m)^{N-i} \tag{1x}$$

Let E(i) be the expected number of requests accepted by the N x N crossbar during a cycle; given that i requests arrived in the cycle. To evaluate E(i), consider the number of ways that i random requests can map to N distinct memory modules; which is N^i. Suppose now that a particular memory module is not requested. Then the number of ways to map i requests to the remaining (N-1) modules is $(N-1)^i$. Thus, $N^i - (N-1)^i$ is the number of maps in which a particular module is always requested. Thus the probability that a particular module is requested is, $[N^i - (N-1)^i]/N^i$. For every memory module, if it is requested, it means one request is accepted by the network for that module. Therefore, the expected number of acceptances is

$$E(i) = \frac{N^i - (N-1)^i}{N^i} \cdot N = \left[1 - \left(\frac{N-1}{N}\right)^i\right] N \tag{2x}$$

The probability P_A, that a request will be accepted by the N x N crossbar is computed as follows.

$$P_A = \frac{\text{expected number of requests accepted}}{\text{expected number of requests arrived.}} = \frac{\sum_{0\le i\le N} E(i)\cdot q(i)}{\sum_{0\le i\le N} i\cdot q(i)}$$

Using (1x) and (2x) both of the above sums can be simplified as follows.

$$\sum_{0\le i\le N} E(i)\cdot q(i) = N - N\left(1 - \frac{m}{N}\right)^N$$

$$\sum_{0\le i\le N} i\cdot q(i) = mN$$

Therefore,

$$P_A = \frac{1}{m} - \frac{1}{m}\left(1 - \frac{m}{N}\right)^N \tag{3x}$$

and the bandwidth $BW = mNP_A$ requests per cycle (4x)

It is interestig to note that,

$$\lim_{N\to\infty} \left(1 - \frac{m}{N}\right)^N = \frac{1}{e^m}$$

(where e is the base of natural logarithm)

The following approximations are good within 1% of the actual value for $N \ge 30$, and within 5% for $N \ge 8$.

$$P_A \simeq \frac{1}{m}\left(1 - \frac{1}{e^m}\right) \tag{5x}$$

$$BW \simeq \left(1 - \frac{1}{e^m}\right) N \text{ requests per cycle} \tag{6x}$$

Note that equation (6x) implies that the bandwidth increases almost linearly with N.

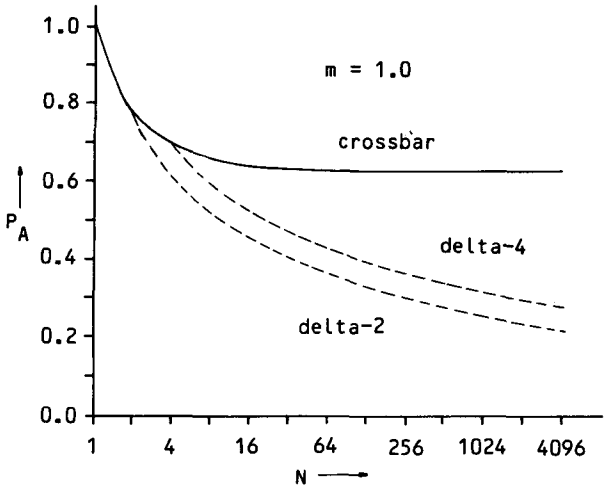

Figure 11. Probability of acceptance of N x N networks.

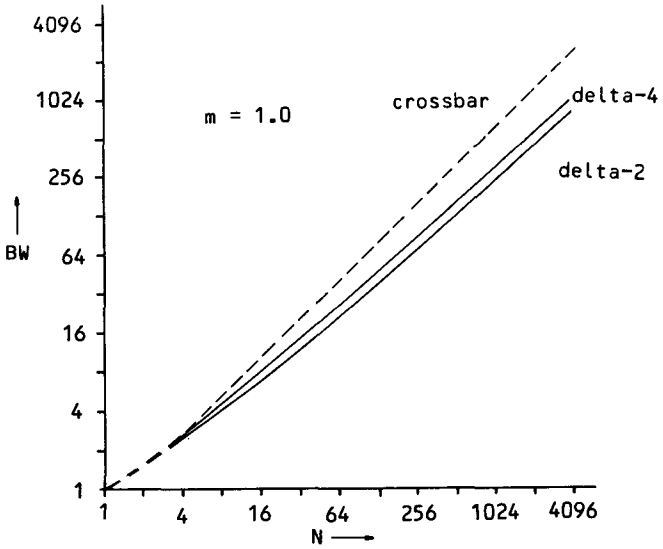

Figure 12. Expected bandwidth of N x N networks.

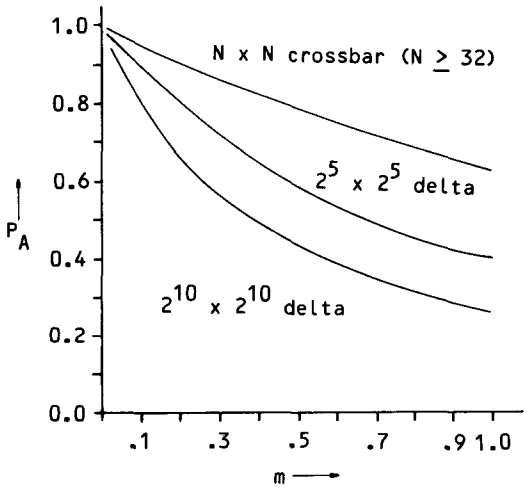

Figure 13. Probability of acceptance vs. mean request rate.

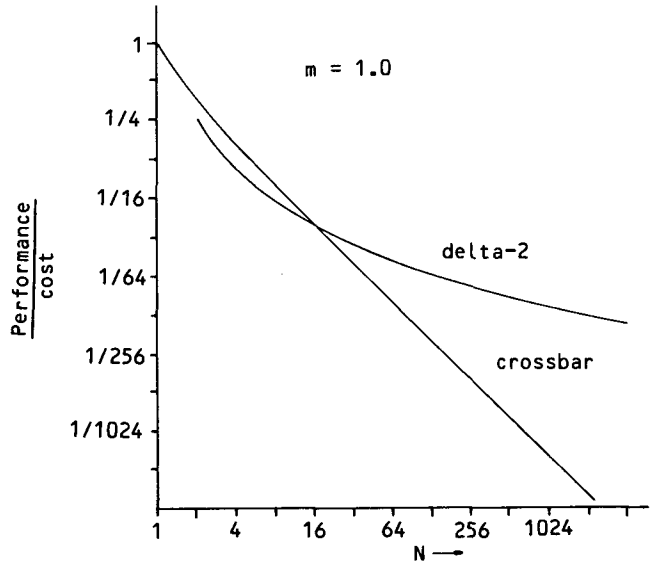

Figure 14. Cost-effectiveness of N x N networks.

VII. Effectiveness of Delta and Crossbar Networks

Using the analysis of sections V and VI we have computed values of the probability of acceptance P_A and the expected bandwidth of several N x N networks. These results are plotted in Figures 11, 12 and 13.

Figure 11 shows the probability of acceptance P_A, for 2^n x 2^n and 4^n x 4^n delta networks and N x N crossbar, when the request generation rate of each processor is m = 1. The curve marked delta-2 is for delta networks using 2 x 2 switches and delta-4 is for delta networks using 4 x 4 switches. The graphs are drawn as smooth curves in this and other figures only for visual convenience, in actuality the values are valid only at specific discrete points. In particular an N x N crossbar is defined for all integers $N \geq 1$, a delta-2 is defined only for $N = 2^n$, $n \geq 1$, and delta-4 is defined only for $N = 4^n$, $n \geq 1$.

Notice in Figure 11 that P_A for crossbar approaches a constant value as was predicted by equation (5x) of the previous section. P_A for delta networks continues to fall as N grows. We have not been able to estimate the limiting value of P_A for delta networks. Figure 12, shows the expected bandwidth, BW as a function of N. The bandwidth is

measured in number of requests accepted per cycle. In all fairness, we must point out that a cycle for a crossbar could be smaller than a cycle for a large delta network. Taking into account fan-in and fan-out constraints, the decoder and arbiter for a N x N crossbar has a delay of $O(\log_2 N)$ gate delays. A $2^n \times 2^n$ delta network has $O(2\log_2 N)$ gate delays, from the analysis of section IV. If the delay is small compared to the memory access time, then the cycle time (the sum of network delay and memory access time) of a crossbar is not too different from that of a delta network. Thus the curves for bandwidth provide a good comparison between networks.

Figure 13 shows P_A as a function of the request generation rate m. The curve for the crossbar is the limiting value of P_A as N grows to infinity. Curves for $N \geq 32$ are not distinguishable with the scale used in that graph.

Finally, the graph of Figure 14 is an indication of cost-effectiveness of delta networks. The cost of a N x N crossbar or delta network is assumed to be proportional to the number of gates required. The constant of proportionality should be the same in both cases, because the degree of integration, modularity and wiring complexity in both cases is more or less the same. For the N x N crossbar, the minimum number of gates required is one per crosspoint per data line. Depending on the assumptions used on fan-in, fan-out, the complexity of the decoder and the arbiter, one can estimate the gate complexity of a crossbar anywhere from one gate to six gates per crosspoint. Let us assume the lowest cost figure of one gate per crosspoint.

The cost of $2^n \times 2^n$ delta network is estimated from the boolean equations of the 2 x 2 module of Figure 9. The number of gates in a 2 x 2 module is 23 gates for the control plus 6 gates per information line. Assuming the number of information lines to be large, the gates for control can be ignored. Thus the gate count of a $2^n \times 2^n$ delta network is $6n2^{n-1}$ per information line because the network has $n2^{n-1}$ modules.

Thus the costs of N x N networks are, kN^2 for crossbar, and $3kN\log_2 N$ for delta ($N = 2^n$), where k is the constant of proportionality. We have used these cost expressions in the computation of performance-cost ratio for Figure 14; the ratio is that of expected bandwidth over cost. Taking this ratio for a 1 x 1 crossbar as unity, the Y-axis of Figure 14 represents the performance over cost relative to a 1 x 1 crossbar. The Y-axis may also be interpreted as bandwidth per gate per information line. Notice that delta network is more cost-effective for network size N greater than 16. If the cost of the crossbar was assumed as 2 or more gates per crosspoint then the curve for the crossbar would shift downward and the effectiveness of delta becomes even more pronounced. However, if one assumed the cycle time of a crossbar half as much as that of a delta network then the curve for crossbar would shift upwards relative to the curve for delta, thus shifting the crossover point of the two curves towards right. Thus depending on the assumptions, the crossover point may move slightly left or right; but in any case, the curves clearly show the effectiveness of delta networks for medium and large scale multiprocessors.

VIII. Concluding Remarks

We have presented in this paper a class of processor-memory interconnection networks, called delta networks, which are easy to control and design, and are very cost-effective. We also presented the combinatorial analysis of delta networks and full crossbars. It is seen that delta networks bridge the gap between a single time-shared bus and a full crossbar. The cost of a N x N delta network varies as $N\log_2 N$ while that of crossbar varies as N^2. Thus delta networks are very suitable for relatively low cost multimicroprocessor systems. If some semiconductor manufacturer will fabricate the 2 x 2 delta modules in large quantities, then the delta networks will become even more affordable and will provide a boost to the construction of many experimental multiprocessor systems.

References

[1] D. H. Lawrie, "Access and Alignment of Data in an Array Processor", IEEE Trans. Comput., Vol. C-24, pp. 1145-1155, Dec. 1975.

[2] M. C. Pease, "The Indirect Binary n-Cube Microprocessor Array", IEEE Trans. Comput., Vol. C-26, pp. 458-473, May 1977.

[3] H. J. Siegel and S. D. Smith, "Study of Multistage SIMD Interconnection Networks", Proc. 5th Annual Symp. on Computer Architecture, pp. 223-229, April 1978.

[4] H. S. Stone, "Parallel Processing with the Perfect Shuffle", IEEE Trans. Comput., Vol. C-20, pp. 153-161, Feb. 1971.

[5] V. E. Benes, Mathematical Theory of Connecting Networks and Telephone Traffic, Academic Press, New York, 1965.

[6] D. Y. Chang, D. J. Kuck and D. H. Lawrie, "On the Effective Bandwidth of Parallel Memories", IEEE Trans. Comput., Vol. C-26, pp. 480-490, May 1977.