Expanded Delta Networks for Very Large Parallel Computers

Brian D. Alleyne
Department of Electrical Engineering
Princeton University
Princeton, New Jersey 08544

Isaac D. Scherson
Department of Information and Computer Science
University of California
Irvine, California 92717

Technical Report #92-02

January 7, 1992

Expanded Delta Networks for Very Large Parallel Computers[*][†]

Brian D. Alleyne
Department of Electrical Engineering
Princeton University
Princeton, New Jersey 08544
(714) 856-7713
alleyne@ics.uci.edu

Isaac D. Scherson
Department of Information and Computer Science
University of California, Irvine
Irvine, California 92717
(714) 856-8144
isaac@ics.uci.edu

Abstract

In this paper we analyze a generalization of the traditional delta network, introduced by Patel [21], and dubbed *Expanded Delta Network* (EDN). These networks provide in general multiple paths that can be exploited to reduce contention in the network resulting in increased performance. The crossbar and traditional delta networks are limiting cases of this class of networks. However, the delta network does not provide the multiple paths that the more general expanded delta networks provide, and crossbars are to costly to use for large networks. The EDNs are analyzed with respect to their routing capabilities in the MIMD and SIMD models of computation.

The concepts of *capacity* and *clustering* are also addressed. In massively parallel SIMD computers, it is the trend to put a larger number processors on a chip, but due to I/O constraints only a subset of the total number of processors may have access to the network. This is introduced as a Restricted Access Expanded Delta Network of which the MasPar MP-1 router network is an example.

Keywords: interconnection networks, delta networks, crossbar, hyperbar, clustering, capacity, MIMD, SIMD.

[*]We gratefully acknowledge Tom Blank and Russ Tuck of Maspar Corporation for sharing with us the inner workings of the Maspar MP-1 router network. Their contributions and discussions were invaluable to this work.

[†]This research was supported in part by the Air Force Office of Scientific Research under grant number AFOSR-90-0144 and NSF under grant number MIP9106949.

1 Introduction

Multistage interconnection networks have been extensively investigated over the past 40 years. Initially, they were used as building blocks for telephone switching networks [7, 5]. Later, they were studied as an alternative to the crossbar and spanning bus for interconnecting processors and memories in multiprocessor systems. Many families of these networks were proposed and studied, including the "Omega network" [14], the "Delta network" [21], and variants of the "Multistage cube" networks [26, 3, 1]. Many of these networks were eventually incorporated into MIMD and SIMD parallel computers. Examples of these include the Maspar MP-1 [18] the IBM RP-3 [22], the NYU Ultracomputer [8] and the GP1000 by BBN Advanced Computers Inc. [4]. This paper analyzes the *Expanded Delta Network* (EDN) which is a generalization of the traditional delta network introduced by Patel [21]. EDNs share the digit controlled routing strategy of delta networks so that no global controller is necessary to set up the switches of the network. However, unlike delta networks, EDNs contain multiple paths (multipath) between any input and output. This fact can be used to reduce conflicts or Non Uniform Traffic Spots (NUTS) [13] that occur within the network.

The concept of capacity (defined later) is similar to the concept of "dilation" [28, 29] in that the networks are "multipath". However the number of wires between stages in a d-dialated network is d times the number of wires of the equivalent stage of an EDN with the same number of inputs, resulting in a much less space efficient network.

In Section 2 the Expanded Delta Network is defined and some of its properties are described. Section 3 deals with the general performance of the EDN. Section 4 expands the analysis to that of MIMD processor memory or processor systems. In Section 5 the EDN is used as a restricted access network [31] in an SIMD environment. Concluding remarks are presented in Section 6.

2 Description of the Expanded Delta Network

This section is divided into three main portions: The characterization of the switch used in the Expanded Delta Networks (EDNs), the characterization of the interconnection permutation between stages, and the question of routing data through the network.

The MP-1 massively parallel processor, produced by Maspar Corporation, uses a unique switch in its router network called a *hyperbar* switch [6]. The generalized version of this switch is the main building block of the Expanded Delta Network (EDN). A detailed analysis of this switch is presented in [17].

The Hyperbar switch is defined as follows:

Definition 1 *A hyperbar, denoted by* $H(a \rightarrow b \times c)$ *is a switch that connects* **a** *inputs labeled* $0, 1, 2, \cdots, a-1$ *to* $b \times c$ *outputs labeled* $0, 1, 2, \cdots, (b \times c) - 1$. *The outputs are labelled such that every group of* **c** *outputs has a label in the range* $[0$ *to* $b-1]$. *There are* **b** *output groups (or buckets) each with capacity* **c**. *A control digit* **d** *of base-b is supplied by each input. This digit indicates which of the* **b** *output groups they are to be connected to. Since each of the* **b** *output groups contains only* **c** *wires, if more than* **c** *inputs request to be connected to a particular output group, exactly* **c** *are accepted and the rest are rejected (See Figure 1).*

The degenerate case $H(a \rightarrow b \times 1)$ is a traditional $a \times b$ crossbar. In Figure 2 we show a $H(8 \rightarrow 4 \times 2)$ hyperbar. In this case, only $\log_2(4) = 2$ bits are needed at each input to determine the appropriate output bucket. A sample switch routing is shown in Figure 2. Note that some the inputs to be discarded since their destination buckets were already full. Assuming that inputs are prioritized according to their input label $(0, 1, 2, \cdots, 7)$, inputs 5 and 7 are discarded.

For simplicity we shall assume that a, b, c are all powers of 2. However, the analysis can easily be expanded to the more general case.

The hyperbar switch is the basic building block for EDNs which are defined as follows:

Definition 2 *An EDN(a,b,c,l) is an* $(l+1)$ *stage interconnection network. The first* $0 \cdots (l-1)$ *stages consist of* $H(a \rightarrow b \times c)$ *hyperbar switches, and the last stage consists of* $c \times c$ *crossbar (or* $H(c \rightarrow c \times 1)$ *) switches. All paths from any input to any output have constant length, and none of the switches have unconnected terminals. All of the output terminals from one stage are connected to input terminals of the next stage.*

An EDN network will have $(a/c)^l c$ inputs and $b^l c$ outputs. At the output of the ith stage $(1 \leq i \leq l)$ there are $(a/c)^{l-i} b^i c$ wires. The ith stage has $(a/c)^{l-i} b^{i-1}$ hyperbars, and the $l+1$ stage has b^l crossbars. Let the switches be named $0, 1, 2, \cdots$ from top to bottom (Figure 3).

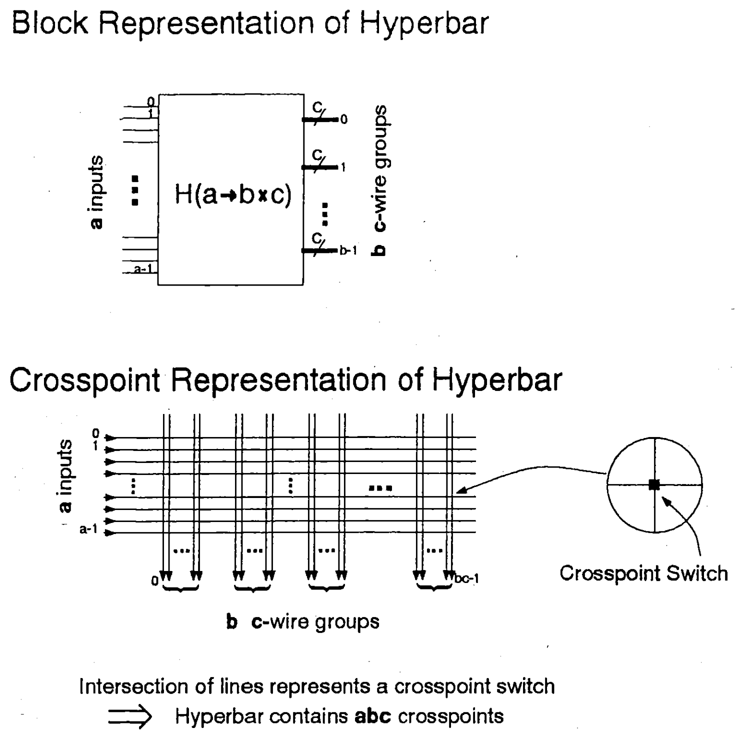
Block Representation of Hyperbar
a inputs
0
1
a-1
H(a→b×c)
c
0
1
b-1
b c-wire groups
Crosspoint Representation of Hyperbar
a inputs
0
1
a-1
0
bc-1
b c-wire groups
Crosspoint Switch
Intersection of lines represents a crosspoint switch
⟹ Hyperbar contains abc crosspoints

Figure 1: A $H(a \to b \times c)$ hyperbar

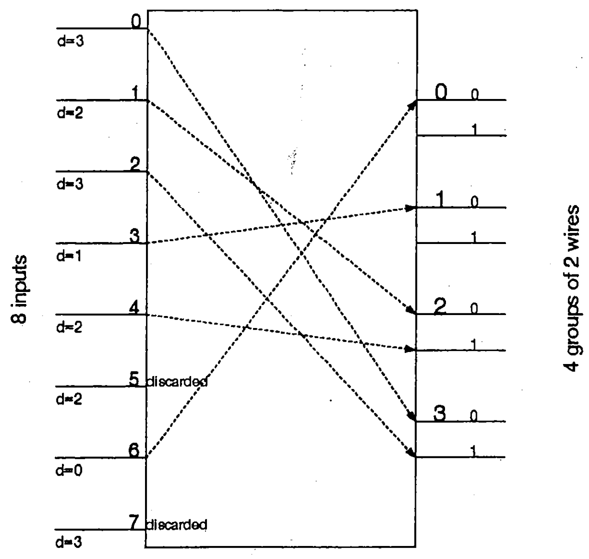

Figure 2: A $H(8 \to 4 \times 2)$ hyperbar

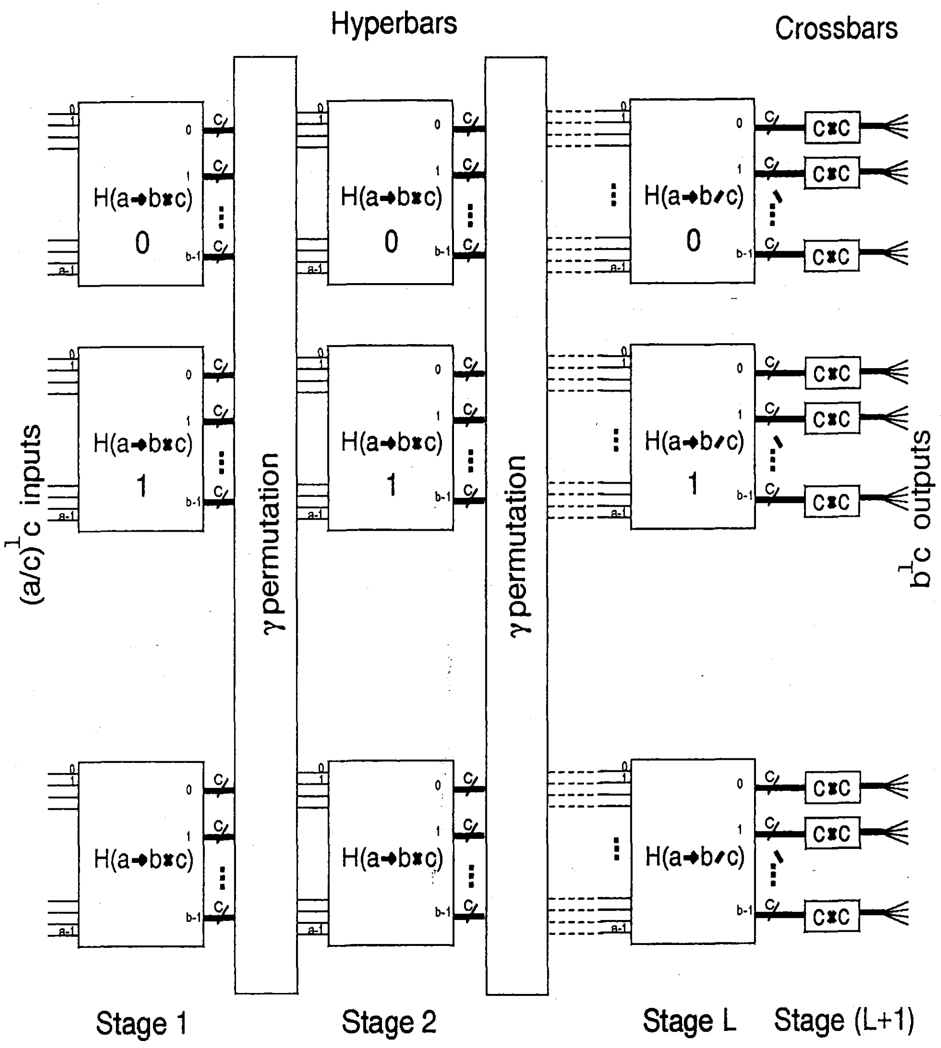

Figure 3: An EDN(a,b,c,l)

To define the connectivity rule between stages we define a γ permutation:

Definition 3 *Permutation $\gamma^n_{j,k}(y)$ is defined on an n-bit label y as follows:*

1) Fix the j least significant bits of the label

2) Left cyclic shift by k the remaining $(n-j)$ bits

This function is related to the "segment shuffle" defined by Lenfant [16]. $\gamma^n_{0,1}(0 \leq i < 2^n)$ is the well-known shuffle of 2^n labels. $\gamma^n_{0,\log_2(q)}(0 \leq i < 2^n)$ is a q-shuffle of 2^n objects defined by Patel [21]. $\gamma^n_{n,0}(0 \leq i < 2^n)$ is the identity permutation.

At the output of stage i and the input of stage $(i+1)$, there are $w_i = (a/c)^{l-i}b^ic$ wires. Let y be any output of stage i, $y \in \{0, 1, 2, \cdots, (a/c)^{l-i}b^ic\}$, and let y be represented by a binary string of length $\log_2((a/c)^{l-i}b^ic)$. Then y is connected to input z of stage $(i+1)$ if and only if

$$z = \gamma^n_{j,k}(y) \text{ where } n = \log_2(w_i),\ j = \log_2(c),\ k = \log_2(a/c). \quad (1)$$

The generalized EDN is shown in Figure 3, and a specific instance is shown in Figure 4. Note that at the lth stage, each of the b^l buckets are sent directly to a $c \times c$ crossbar.

Before proving that an EDN is indeed "connected" let us first illustrate how routing is performed on EDNs. At every source a $(l \times \log_2(b) + \log_2(c))$ destination tag is used for routing. At each hyperbar stage, $log_2(b)$ bits are used for routing, and at the final $c \times c$ crossbar stage, $\log_2(c)$ bits are used. Let the destination tag be written as $D = d_{l-1}d_{l-2}\cdots d_0x$ where the $d_{i's}$ are digits in a base-b system, and x is a digit in a base-c system. After the destination tags pass through the network, then some sources are connected to some destinations. At this point data is transmitted through the network.

Routing in the EDN is performed as follows:

1. At stage i, $(1 \leq i \leq l)$, the digit d_{l-i} of the destination tag $D = d_{l-1}d_{l-2}\cdots d_0x$ determines which of the output buckets of a hyperbar a message should be connected to. For example, if a particular message has $d_{l-i} = 1$ at stage i, then that message should be routed to the $1st$ output bucket of the hyperbar it is passing through. It does not matter on which of the c wires of the output bucket the message is placed. Thus, there are c possible choices of output at every stage. If more than c messages require to be routed to any particular bucket, only c

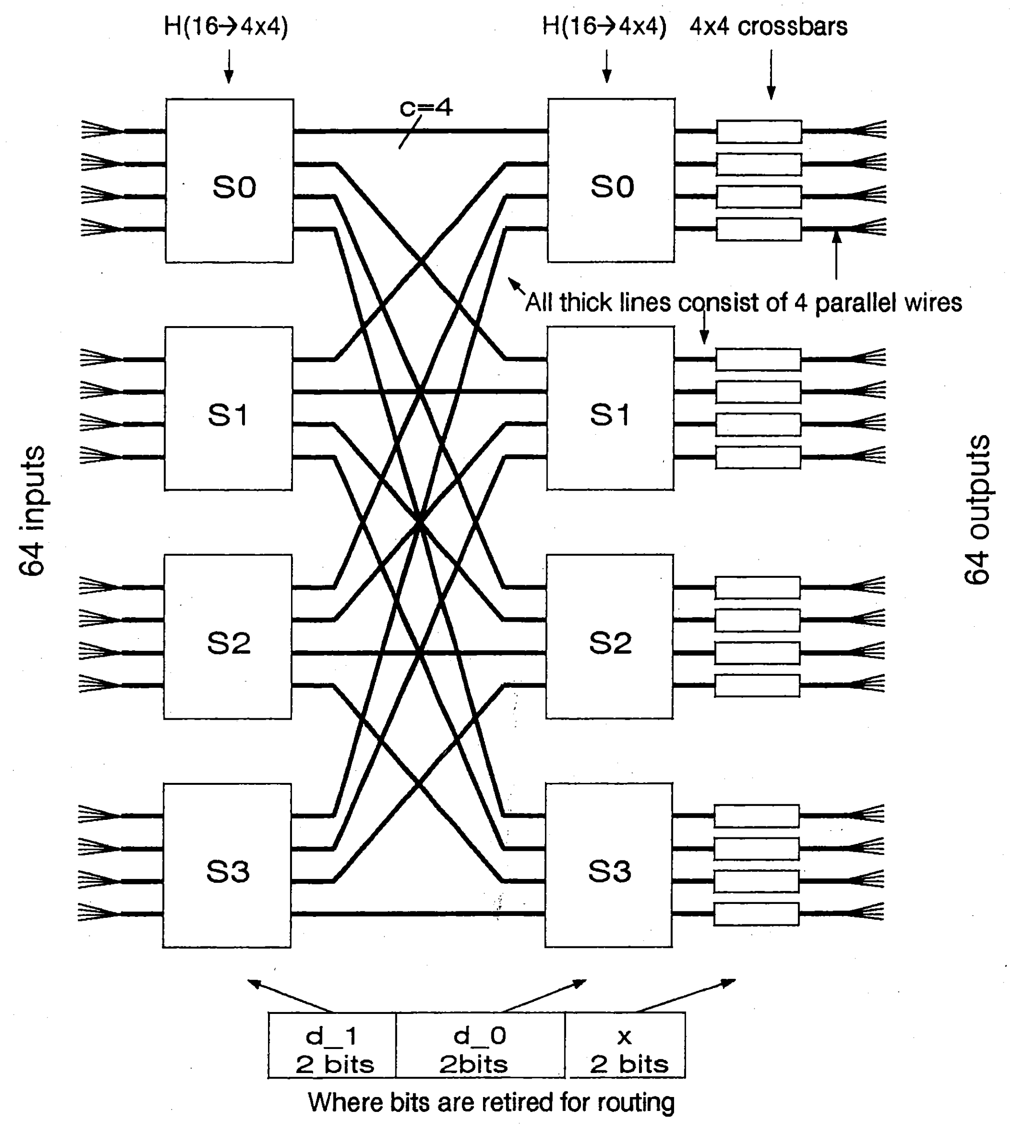

Figure 4: An EDN(16,4,4,2)

are accommodated, and the rest discarded. Since each d_{l-i} is used once during the routing and are never considered again, we liken this to "retiring" and say that bits d_{l-i} of D are "retired" at stage i.

2. At stage $(l+1)$, the messages are inputs to a $c \times c$ crossbar. At this point the digit x of the routing tag $D = d_{l-1}d_{l-2}\cdots d_0x$ is used to determine to which output the message is to be routed. Similarly, we say that bits x of D are "retired" at stage $l+1$.

Lemma 1 *An EDN(a,b,c,l) can connect any source S to any destination $D = d_{l-1}d_{l-2}\cdots d_0x$ by retiring d_{l-i} of D at stage i ($1 \leq i \leq l$) and by retiring x at stage $l+1$.*

Proof: A similar approach to that used by Patel [21] to prove that a delta network can connect any source to any destination is used for this proof.

Consider a source S that needs a connection to destination D. The destination address is represented as $d_{l-1}d_{l-2}\cdots d_0x$, where the $d_{i's}$ are digits of a base-b system, and x is a digit in a base-c system. Let S be represented as $s_{l-1}s_{l-2}\cdots s_0x'$ where the $s_{i's}$ are digits of a base-(a/c) system and x' is a digit in a base-c system. At each stage, let the switches be named $0, 1, 2, \cdots$ from top to bottom. At the input and output of each stage, let the lines be named $0, 1, 2, \cdots$ from top to bottom.

At the input to the network, S is connected to hyperbar number $\lfloor S/a \rfloor$. The input is now routed to bucket d_{l-1}, since d_{l-1} is used for routing at stage 1. Thus, the source is connected to line

$$L_1 = (\lfloor S/a \rfloor b + d_{l-1})c + K_1$$

where $0 \leq K_1 < c$. K_1 cannot be determined since we do not know to which output of bucket d_{l-1} the source S will be switched.

In order to compute $\lfloor S/a \rfloor$, we make the observation that the least significant digit of S is x' which is a digit in base-c, and thus consists of $log_2(c)$ bits, and the next significant digit, s_0, is a digit in base-(a/c) which consists of $log_2(a/c)$ bits. Thus s_0x' has $log_2(a/c) + log_2(c) = \log_2(a)$ bits. $\lfloor S/a \rfloor$ is now computed by right-shifting S by $\log_2(a)$ bits, which is equivalent to dropping s_0x' from S yielding $s_{l-1}s_{l-2}\cdots s_1$.

And so

$$L_1 = ((s_{l-1}s_{l-2}\cdots s_1)b + d_{l-1}) \times c + K_1$$

Now we have to compute L_1', the input to stage 2. Using the definition of the interconnection between stage 1 and stage 2, we get a line number:

$$L_1' = \gamma^{\log_2(w_1)}_{\log_2(c),\log_2(a/c)}(L_1)$$

Now, $((s_{l-1}s_{l-2}\cdots s_1)b+d_{l-1})c$ is strictly greater than c, and K_1 is strictly less than c. This means that the γ function which fixes the least $\log_2(c)$ bits leaves the K_1 portion of L_1 unchanged, and cyclicly shifts the bit string $((s_{l-1}s_{l-2}\cdots s_1)b + d_{l-1})$ by $log_2(a/c)$, giving $((s_{l-2}s_{l-3}\cdots s_1)ab/c + s_{l-1} + d_{l-1}a/c)$. Collecting these terms:

$$L_1' = (s_{l-2}s_{l-3}\cdots s_1)ab + d_{l-1}a + s_{l-1}c + K_1$$

which is the input to stage 2. This line is connected to switch $\lfloor L_1'/a \rfloor$ of stage 2 and then routed to output $d_{l-2}c + K_2$ (where $0 \le K_2 < c$) of that switch and becomes line L_2 where

$$L_2 = (\lfloor L_1'/a \rfloor b + d_{l-2})c + K_2$$

Now

$$\begin{aligned} \lfloor L_1'/a \rfloor &= \lfloor((s_{l-2}s_{l-3}\cdots s_1)ab + d_{l-1}a + s_{l-1}c + K_1)/a\rfloor && (0 \le K_l < c) \\ &= (s_{l-2}s_{l-3}\cdots s_1)b + d_{l-1} + \lfloor(s_{l-1}c + K_1)/a\rfloor \\ &= (s_{l-2}s_{l-3}\cdots s_1)b + d_{l-1} \end{aligned}$$

since $(s_{l-1}c + K_1) < a$. Substituting into L_2 we get

$$L_2 = ((s_{l-2}s_{l-3}\cdots s_1)b^2 + d_{l-1}b + d_{l-2})c + K_2$$

In general, after the ith stage, the input to the network is line

$$L_i = ((s_{l-i}s_{l-i-1}\cdots s_1)b^l + d_{l-1}b^{i-1} + d_{l-2}b^{i-2} + \cdots + d_{l-i})c + K_i \qquad (0 \le K_i < c)$$

Since all $d_{i's}$ are digits in base-b, this expression can be shortened to:

$$L_i = ((s_{l-i}s_{l-i-1}\cdots s_1)b^l + d_{l-1}d_{l-2}\cdots d_{l-i})c + K_i$$

In particular, at the lth stage

$$L_l = (d_{l-1}d_{l-2}\cdots d_0)c + K_l$$

At the final stage the message is sent to the $\lfloor L_l/c \rfloor$ crossbar of the final stage, and routing in the crossbar is effected by the last digit x in base-c. So,

$$\begin{aligned} L_{out} &= (\lfloor L_l/c \rfloor)c + x \qquad\qquad (0 \leq K_l < c) \\ &= (\lfloor((d_{l-1}d_{l-2}\cdots d_0)c + K_l)/c \rfloor)c + x \\ &= (d_{l-1}d_{l-2}\cdots d_0)c + x \\ &= d_{l-1}d_{l-2}\cdots d_0 x \end{aligned}$$

Thus the input is routed to the required destination $\Box Q.E.D.$

Theorem 1 *An EDN is always connected.*

Proof: Follows constructively from Lemma 1 $\Box Q.E.D.$

Corollary 1 *A renaming of the inputs of an EDN or a permutation of its inputs does not prevent a source from connecting to destination* $D = d_{l-1}d_{l-2}\cdots d_0 x$.

Proof: By Theorem 1, an EDN(a,b,c,l) is always connected. If a path exists we can connect source to destination $D = d_{l-1}d_{l-2}\cdots d_0 x$ by Lemma 1, irrespective of where the message originated. Thus renaming the inputs, or permuting the inputs, only puts $D = d_{l-1}d_{l-2}\cdots d_0 x$ onto a different input to the network. From this input it is routed according to the routing algorithm to the appropriate destination. $\Box Q.E.D.$

Corollary 2 *If the bits of* $D = d_{l-1}d_{l-2}\cdots d_0 x$ *are retired in a different order, say* d'_{l-i} *at stage* i, *then the source with destination tag* $d_{l-1}d_{l-2}\cdots d_0 x$ *will be routed to destination* $D' = d'_{l-1}d'_{l-2}\cdots d'_0 x'$.

Proof: The above statement is equivalent to saying that the bits of $d_{l-1}d_{l-2}\cdots d_0 x$ are reordered such that $F(d_{l-1}d_{l-2}\cdots d_0 x) = d'_{l-1}d'_{l-2}\cdots d'_0 x'$ before being fed to the network. In this case since the routing tag $d'_{l-1}d'_{l-2}\cdots d'_0 x'$ is applied to the network, the input will be routed to $D' = d'_{l-1}d'_{l-2}\cdots d'_0 x'$ by Theorem 1. But $D' = F(d_{l-1}d_{l-2}\cdots d_0 x) = F(D)$. Thus to retire the

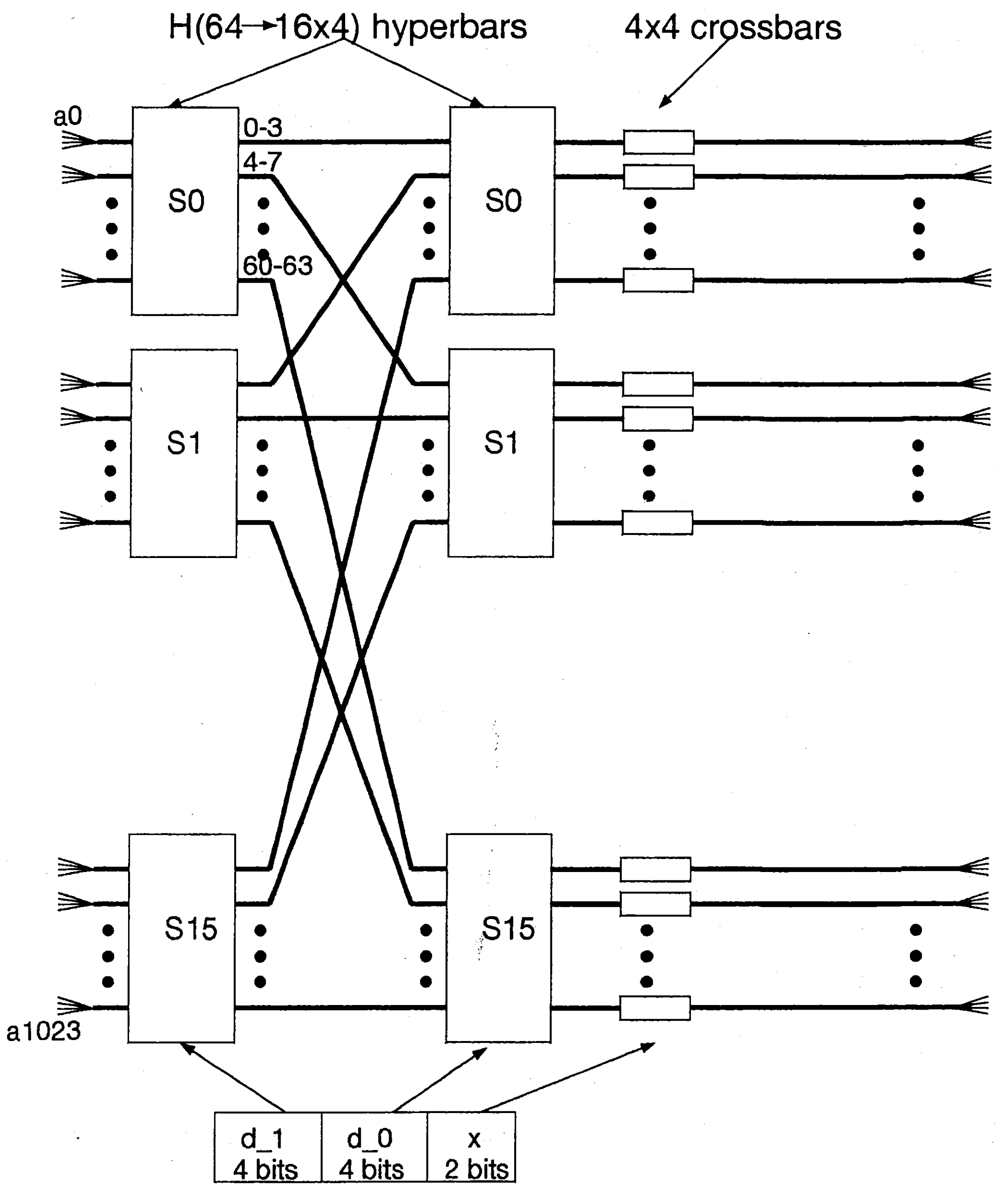

Figure 5: An EDN(64,16,4,2)

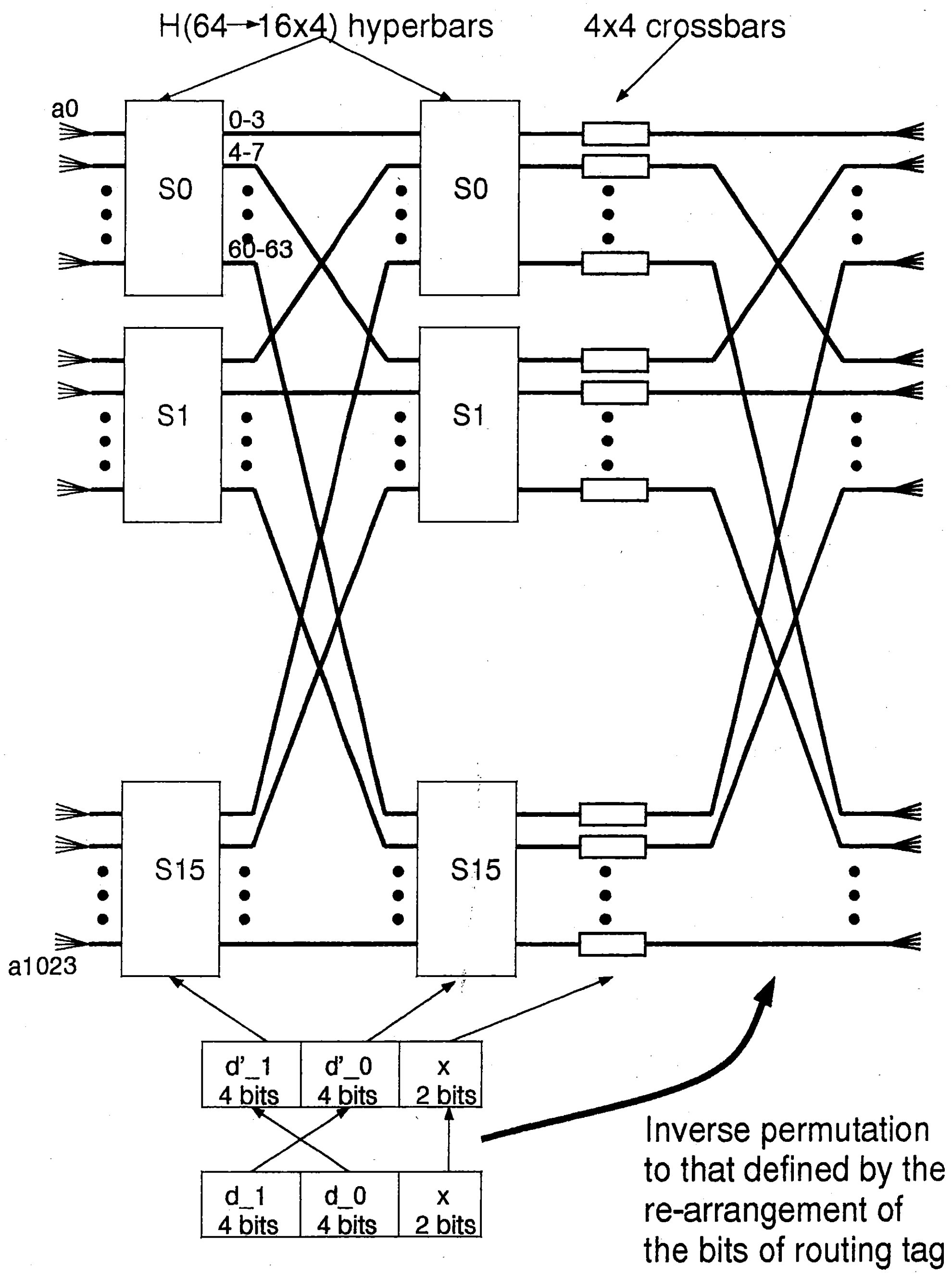

Figure 6: An EDN(64,16,4,2) modified to perform the identity permutation

bits in different order, while preserving the destination $D = d_{l-1}d_{l-2}\cdots d_0x$, the permutation F^{-1} must be performed at the output of the network. $\square Q.E.D.$

To illustrate the usefulness of Corollary 2, consider the EDN(64,16,4,2) shown in Figure 5. This network is incapable of performing the identity permutation in one pass. However, by retiring the bits of the routing tag in a different order, and then adding an additional permutation stage to compensate, the modified EDN in Figure 6 is obtained. It should be noted that these networks will perform identically in the average case, while very differently for *specific* permutations.

Theorem 2 *An EDN(a,b,c,l) has c^l different paths from any input to any output.*

Proof: At each stage i, digit d_{l-i} in $D = d_{l-1}d_{l-2}\cdots d_0x$ determines to which bucket the source is switched. However, it can be put onto any one of the c wires of the bucket, ie. the source can be put onto wires $d_{l-i}c, d_{l-i}c+1, \cdots, d_{l-i}c+(c-1)$. Thus, there are c alternate paths that the source can be switched to at each stage. Since this occurs in l stages, there are c^l possible paths that the source can take to any one output. $\square Q.E.D.$

There are two special cases of EDNs worthy of mention. An EDN(a,b,1,1) is an $a \times b$ crossbar. An EDN(a,b,1,l) is an $a^l \times b^l$ delta network [21]. In both of these cases, $c = 1$ and so by Theorem 2 there is a unique path from any input to any output.

3 General Analysis of EDNs

3.1 Cost of EDNs

The number of crosspoint switches ($C_s(a,b,c,l)$) required to build the network is a possible measure of the cost of the EDN(a,b,c,l). This is a reasonable measure since the number of crosspoints give an idea of the layout area necessary to realize the network. Thus, an $a \times b$ crossbar containing ab crosspoint switches has an associated cost of ab, and an $H(a \rightarrow b \times c)$ hyperbar with abc crosspoint switches has a cost of abc. The EDN(a,b,c,l) consists of l stages of hyperbars, and one stage of crossbars. For each stage i ($1 \leq i < l$), there are $(a/c)^{l-i}b^{i-1}$ $H(a \rightarrow b \times c)$ hyperbars, and in the final stage there are b^l $c \times c$ crossbars. In total, there are

$$\sum_{i=1}^{l}(a/c)^{l-i}b^{i-1} = \frac{(a/c)^l - b^l}{(a/c) - b} \quad (a/c) \neq b$$

$$= lb^{l-1} \qquad (a/c) = b$$

hyperbars and b^l crossbars. Thus the crosspoint switch cost of an EDN(a,b,c,l) is

$$\begin{aligned} C_s(a,b,c,l) &= \tfrac{(a/c)^l - b^l}{(a/c)-b} abc + b^l c^2 \quad (a/c) \neq b \\ &= lb^{l+1}c + b^l c^2 \qquad (a/c) = b \end{aligned} \tag{2}$$

Another measure of cost of the EDN(a,b,c,l) is the wire cost $C_w(a,b,c,l)$, that is the number of wires required to connect all the hyperbars and crossbars that constitute the network. This is important, since this cost provides an estimate of PC board area, the number of pins and, in some cases, the number of connections needed across the backplane. The number of wires between stage i and $i+1$ is $(a/c)^{l-i}b^i c$. Thus, the total number of wires between stages is

$$\begin{aligned} \sum_{i=1}^{l} (a/c)^{l-i} b^i c &= \tfrac{(a/c)^l - b^l}{(a/c)-b} bc \quad (a/c) \neq b \\ &= lb^l c \qquad (a/c) = b \end{aligned}$$

The number of inputs and outputs to the network is $(a/c)^l$ and $b^l c$ respectively. A wire is counted for each of these. Thus, the wire cost of an EDN(a,b,c,l) is

$$\begin{aligned} C_w(a,b,c,l) &= \tfrac{(a/c)^l - b^l}{(a/c)-b} bc + (a/c)^l c + b^l c \quad (a/c) \neq b \\ &= (l+2) b^l c \qquad (a/c) = b \end{aligned} \tag{3}$$

3.2 Performance of EDN's

Theorem 1 showed that an EDN(a,b,c,l) is capable of routing any input to any output. However, if many inputs require to be routed simultaneously, there is the possibility that some of the inputs will not be routed. This occurs due to two reasons:

1. Two or more inputs may contend for the same output. In this case all but one of these inputs will be blocked (not accepted).

2. Even if there is no contention for an output, an input may be blocked as it makes its way through the network itself. This will not occur only in the special case where the EDN is a crossbar. As a result, even if the inputs form a permutation (in which case a crossbar would

be able to route all inputs), in general there is no guarantee that all the inputs will be routed by the EDN.

A cycle is defined as the time required for a request at any input to propagate through the network to an output (if not blocked), plus the time for the corresponding message to propagate through the network. It is assumed that the network is circuit-switched, and so there are no buffers or queues in the network. At the beginning of each cycle, the network attempts to accommodate all the requests presented at the inputs. Some of the requests are blocked, and so the number of requests actually satisfied is a fraction of the requests issued. The probability of acceptance P_A is defined as the ratio of the expected number of requests satisfied per cycle to the expected number of requests generated per cycle. We will proceed to derive P_A for EDNs.

For the purpose of analysis, the following assumptions are made about the nature of the requests generated:

1. inputs are uniformly and independently distributed over the outputs. Thus at each cycle, the probability that any input should be connected to a particular output is the same.
2. At the beginning of each cycle, the probability that there is a request on an input line is r.
3. The requests which are blocked are ignored, and do not affect the requests generated at the next cycle. ie. the requests generated at each cycle are independent of the inputs blocked in previous cycles.

Theorem 3 *If the inputs to the network are uniformly and independently distributed over the outputs, then the inputs to every hyperbar in the network are uniformly and independently distributed over the output buckets of the hyperbars at stage i $(1 \leq i \leq l)$, and the inputs to the crossbars at stage $l+1$ are uniformly distributed over their outputs.*

Proof: Each stage of the EDN is controlled by a distinct digit of the destination tag. In particular, stage i $(1 \leq i \leq l)$ is controlled by d_i, and stage $l+1$ is controlled by c (destination tag represented as in Section 2). The inputs to the network are uniformly and independently distributed over the outputs, which implies that the destination tags are uniformly and independently distributed. This in turn implies that the digits d_i and c are uniformly and independently distributed. Since the digits d_i are used as routing tags determine which output bucket of the hyperbar the message is routed to, then the inputs to the hyperbars are uniformly and independently distributed over the output buckets of the hyperbar. Since the digit c is used as as a routing tag to the crossbars of the network, then the inputs to the crossbars are uniformly and independently distributed. $\Box Q.E.D.$

We will now derive an expression for P_A using Theorem 3. Let us consider a hyperbar $H(a \rightarrow b \times c)$, in which the requests are independently and uniformly distributed over the output buckets. For each of the a inputs of the hyperbar, the probability that there is a request is r. The probability that a request is destined for any of the b output buckets is $1/b$. Thus the probability that a request originates on an input line and is destined for a particular output bucket is r/b. Given that there are a inputs, each with probability r/b of requesting an output bucket,

$$\text{the probability of exactly } n \text{ requests for any bucket} = \begin{cases} \begin{pmatrix} a \\ n \end{pmatrix} \left(\frac{r}{b}\right)^n \left(1-\frac{r}{b}\right)^{a-n} & \text{for } n \leq a \\ 0 \text{ for } n > a \end{cases}$$

Since each bucket has a capacity of c, requests beyond c will be discarded. Thus the expected number of requests accepted per bucket is

$$E(r) = \sum_{n=1}^{c} n \begin{pmatrix} a \\ n \end{pmatrix} \left(\frac{r}{b}\right)^n \left(1-\frac{r}{b}\right)^{a-n} + \sum_{n=1}^{c} c \begin{pmatrix} a \\ n \end{pmatrix} \left(\frac{r}{b}\right)^n \left(1-\frac{r}{b}\right)^{a-n}$$

which simplifies to:

$$c\left(1-\left(1-\frac{r}{b}\right)^a\right) + \sum_{n=1}^{c}(n-c) \begin{pmatrix} a \\ n \end{pmatrix} \left(\frac{r}{b}\right)^n \left(1-\frac{r}{b}\right)^{a-n}$$

Thus the probability that there is a request at an output of the hyperbar is $E(r)/c$.

Using Theorem 3, if r_{in} is the request rate at the inputs of any stage of hyperbars and r_{out} is the request rate at the outputs, then $r_{in} = E(r_{out})$. In particular, $r_{i+1} = E(r_i)/c$ for $0 \leq i < l, r_0 = r$. r_l is the input to the $c \times c$ crossbar stage of the network. By Theorem 3, these inputs are uniformly and independently distributed over the outputs of the crossbar. Since the probability of a request at every input is r_l, and by Theorem 3, these inputs are uniformly and independently distributed over the outputs of the crossbar, the probability that there will *not* be a request at a particular output by any of the c inputs is $(1 - r_l/c)^c$. So the probability that a message was routed is $1 - (1 - r_l/c)^c = r_{final}$.

The probability of acceptance P_A is defined as the ratio of the expected number of requests routed per cycle ((number of outputs $\times r_{final}) = b^l c \times r_{final}$) to the expected number of requests generated per cycle ((number of inputs $\times r) = (a/c)^l c \times r$). Thus

$$P_A(r) \; = \tfrac{b^l c \times r_{final}}{(a/c)^l c \times r} \; = \left(\frac{bc}{a}\right)^l \frac{r_{final}}{r} \tag{4}$$

$$\begin{aligned}
\text{where } r_0 &= r \quad \text{and for } 0 \le i < l \\
r_{i+1} &= \left(1 - \left(1 - \frac{r_i}{b}\right)^a\right) + \sum_{n=1}^{c}\left(\frac{n}{c} - 1\right)\begin{pmatrix} a \\ n \end{pmatrix}\left(\frac{r_i}{b}\right)^n\left(1 - \frac{r_i}{b}\right)^{a-n} \\
r_{final} &= 1 - (1 - r_l/c)^c
\end{aligned}$$

In Figures 7,8 plots are presented which compare the performance of various EDNs. The performance of a crossbar network is also included as reference. In Figure 7, all families EDNs generated with 8 inputs 8 outputs hyperbars are featured. As expected, the EDN(8,8,1,*) which correspond to the family of delta networks performs the worse. In addition, as the capacity is increased, the performance of the networks improves. The performance of the family of EDNs generated by the 16 inputs 16 outputs hyperbars also performs better than the family of EDNs generated by the 8 inputs 8 outputs hyperbars.

3.2.1 Permutation Routing

Let us now assume that the input requests to the EDN form a permutation on the outputs of the network.

Lemma 2 *If the input requests to an EDN form a permutation, there will be no blocking at the final two stages.*

Proof: Since the input requests form a permutation, there will never be a contention for an output of the network. The outputs of the last stage switches are also outputs of the network, and so no contention will occur at these switches.

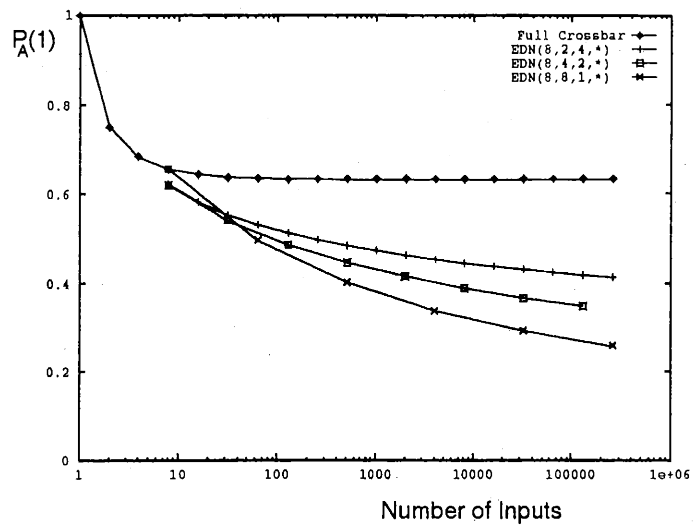

Figure 7: Performance of EDNs with 8 input/output hyperbars

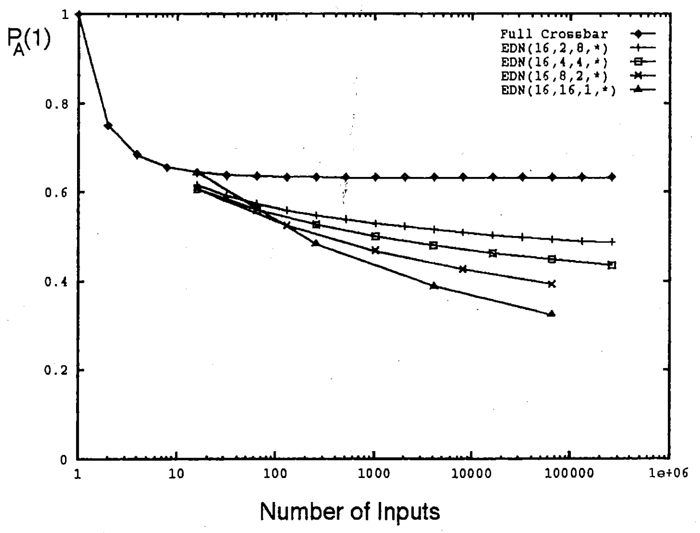

Figure 8: Performance of EDN's with 16 input/output hyperbars

Each of the b output groups of the second-to-last stage is connected directly to a $c \times c$ crossbar switch. This $c \times c$ crossbar is connected directly to c outputs of the network. Therefore, if a permutation is being routed, there will never be more than c requests for any $c \times c$ crossbar. Thus there will be never more than c requests to each of the b output groups of the second-to-last stage, and so all requests can be accommodated at the second-to-last stage. Thus there are no conflicts in the last two stages. $\Box Q.E.D.$

Let us denote the probability of acceptance P_A in the special case where the inputs form a permutation as P_{Ap}. P_{Ap} can be derived from P_A by modifying the equation for P_A to take into account that there will be no blocking in the last two stages by Lemma 2. Thus

$$P_{Ap}(r) = \frac{b^{l-1} c \times r_{final}}{(a/c)^{l-1} c \times r} = \left(\frac{bc}{a}\right)^{l-1} \frac{r_{l-1}}{r} \tag{5}$$

$$\begin{aligned} \text{where } r_0 &= r \quad \text{and for } 0 \leq i < l-2 \\ r_{i+1} &= \left(1 - \left(1 - \frac{r_i}{b}\right)^a\right) + \sum_{n=1}^{c} \left(\frac{n}{c} - 1\right) \left(\begin{array}{c} a \\ n \end{array}\right) \left(\frac{r_i}{b}\right)^n \left(1 - \frac{r_i}{b}\right)^{a-n} \end{aligned}$$

In the following two sections we turn our attention to computing systems in which EDNs are likely to be imbedded, namely MIMD and SIMD machines.

4 Analysis of the EDNs in MIMD Computers

In this section a processor-memory multiprocessor system is assumed. Examples of such systems are the Cedar System [12] and the NYU Ultracomputer [9]. These systems can support a maximum of 1024 and 256 processors respectively. It is further assumed that each node has direct access to the network.

A multiprocessor system now considered in which the processors share a main memory through an EDN(a,b,c,l). Each processor is connected to an input port of the network, and each memory module to an output port. At each cycle, a processor generates a request to any one of the output modules with probability r. Such a system is shown in Figure 9. Under the assumptions of Section 3, the expected bandwidth of the system is given by Equation(4), depending on the request rate r

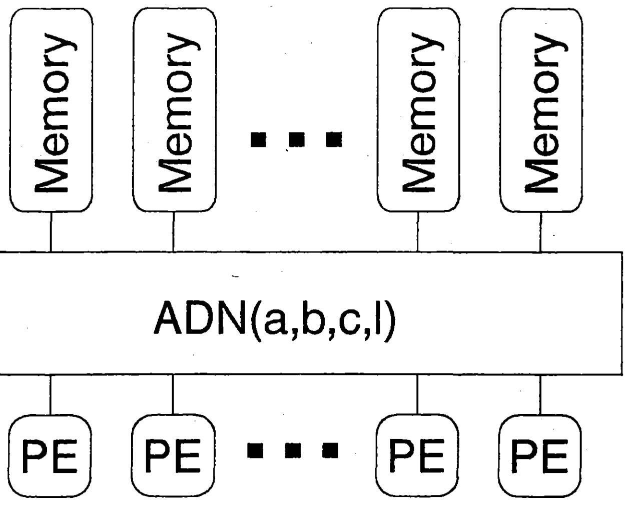

Figure 9: A Multiprocessor System

of the processors. However, a rejected request is generally not discarded, but submitted again on the following cycle, until it is satisfied. This causes a net increase in request rate to the network, and hence a decrease in network performance. We will now consider the magnitude of this effect on network performance using the method described in [11].

Processors that have to resubmit their requests are considered blocked, since it is reasonable to assume that they have to wait for the requested data in order to continue processing. At any given time processors can be in one of two states, active (A) or waiting (W). Let q_A and q_W be the steady state probabilities that a processor is active or waiting respectively, and $P'_A(r)$ be the steady state probability of acceptance of the network. Then the system can be described by the Markov graph of Figure 10.

We will also assume that the resubmitted requests along with the new requests address the memory modules uniformly, thus the assumption that the requests are uniformly and independently distributed is still valid. Solving for q_A and q_W we obtain

$$q_A = \frac{P'_A(r)}{r + P'_A(r) - rP'_A(r)} \tag{6}$$

$$q_W = \frac{r(1 - P'_A(r))}{r + P'_A(r) - rP'_A(r)} \tag{7}$$

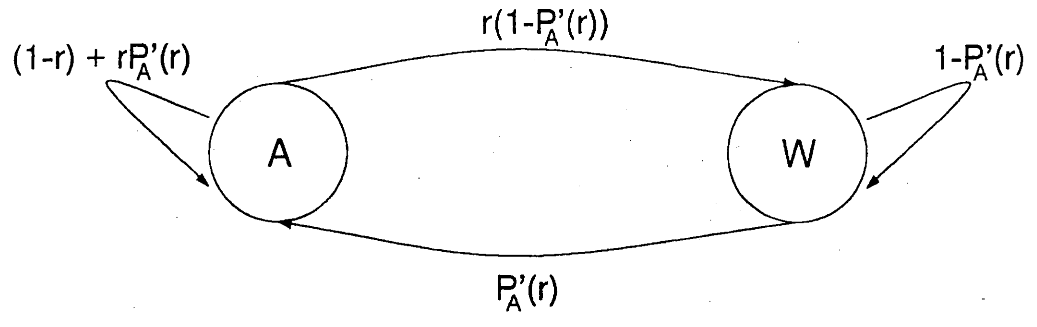

Figure 10: Markov graph for computing q_A and q_W

The request rate to the network is now r', where

$$r' = rq_A + q_W = \frac{r}{r + P'_A(r) - rP'_A(r)} \tag{8}$$

and we have

$$P'_A(r) = P_A(r') = P_A\left(\frac{r}{r + P'_A(r) - rP'_A(r)}\right) \tag{9}$$

where $P_A(r)$ is given by Equation 4. $P'_A(r)$ can be computed iteratively from Equation 9 by computing the following recursion until conversion:

$$\begin{aligned} P'^{n+1}_A(r) &= P_A\left(\frac{r}{r + P'^n_A(r) - rP'^n_A(r)}\right) \\ &\quad \text{with starting condition } P'^0_A(r) = P_A(r) \end{aligned} \tag{10}$$

from which r', q_A and q_W can be determined using Equation 8.

In Figure 4 we see the impact that resubmitting rejected requests has on the network performance in two typical cases.

The efficiency of the shared memory MIMD system in which processors share a main memory through an EDN(a,b,c,l) over an MIMD system in which any memory request will always be satisfied is given by

$$\frac{q_A}{q_A + a_W} = q_A \tag{11}$$

The extension of the above model to processor-processor interconnection in MIMD systems is quite straightforward, and is not expanded upon in this paper.

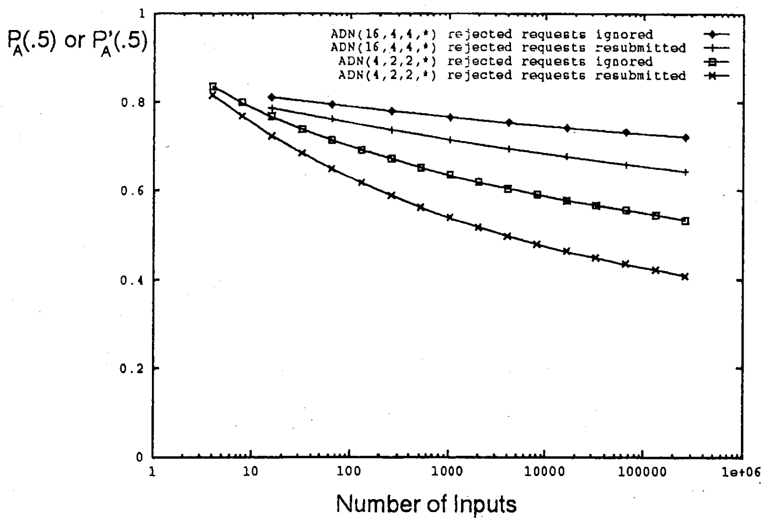

Figure 11: Effect of resubmitting rejected requests on P_A in EDNs

5 Analysis of the EDNs in SIMD Computers

Currently available Massively Parallel computers such as Thinking Machine's CM-x [10], MasPar's MP-1 [18] and AMT's DAP [20] have proven the feasibility of systems in which there are more than 4K processing elements (PEs). Interprocessor communication can be performed in one of two methods, each of which have their own dedicated communication network. Local communication is supported by a mesh like interconnection structure, while global communication is generally performed by a generalized router. The router of the CM is based on a hypercube structure, while that of the MasPar is based on the restricted delta network described in this paper. In MIMD architectures different processors may require communication at different times, and so the "goodness" of the network is based on degree, diameter and bandwidth. However, in an SIMD system all (or at least a good portion) of the processors usually want to communicate at the *same* time. Hence the goal of the router is to route an arbitrary permutation in a reasonable time.

The Connection Machine CM-1 [10] and the Maspar MP-1 [18] have chips containing 16 or 32 processing elements respectively. With the ability to pack more and more processing elements per processing chip, the I/O bottleneck between processing elements and the interconnection network is only aggravated. Already systems are available with 64K PEs. As the number of processing elements continues to grow, it will most certainly be impractical (or impossible) to build interconnection networks where the network size is equal to the number of PEs. One solution to this problem is clustering which is used in the architecture of the MasPar MP-1 system, in thinking Machine's CM-x, and the proposed P^3 system [24, 2, 25]. In clustering, a restricted access network is used where a group of processing elements (a cluster) ,as opposed to a single processing element, has access to the network at any given time. This has been studied in the case where the interconnection network is a crossbar, Clos or Benes Network[31]. Much research has been done on permutation routing on multistage networks [5, 14, 15, 27] as well as on static networks [9, 19, 23, 30] but all these research efforts assume that the network size (i.e., the number of its input terminals) is equal to the number of processors in the system.

We now generalize our analysis of EDNs to systems which incorporate clustering. We will compare the performance of the proposed augmented delta network with that of a crossbar in a restricted access system when performing arbitrary permutations. Random permutations occur in SIMD in cases where the communications are data dependent.

5.1 Restricted Access EDNs

We refer to a restricted access augmented delta network and the associated processing elements as a RA-EDN system. The $RA-EDN$ system consists of p clusters and an interconnection network of size p. Since the number of inputs and outputs of the EDN are the same, the EDN used can be represented as EDN(bc,b,c,l), with $p = b^l c$. Each cluster has q processing elements, a single input port (I) and a single output port (O). For convenience, the clusters are labeled $0, 1, ..., p-1$ and the input and output ports of cluster i are denoted I_i and O_i and are assumed to be the input terminal i and output terminal i of the network, respectively. The processors in each cluster are locally labeled $0, 1, ..., q-1$. Thus, every processor in the system is globally labeled with two digits $\overline{xy}$ indicating that it is processor y in cluster x, where $0 \leq x \leq p-1$ and $0 \leq y \leq q-1$. In decimal notation, the processors are labeled $0, 1, ..., N-1$, where $N = p \times q$ and the decimal label of processor $\overline{xy}$ is $xp + y$. A $RA-EDN$ system so parameterized and labeled is denoted $RA-EDN(b, c, l, q)$ (Figure 12).

Routing a permutation f of the set $S_N = \{0, 1, ..., N-1\}$ in a system of N processors $(0, 1, ..., N-1)$ consists of delivering a message from processor i to processor $f(i)$ for every $i = 0, 1, ..., N-1$. In RA-EDN(b,c,l,q), at most one message from each cluster can be sent at every network cycle (a

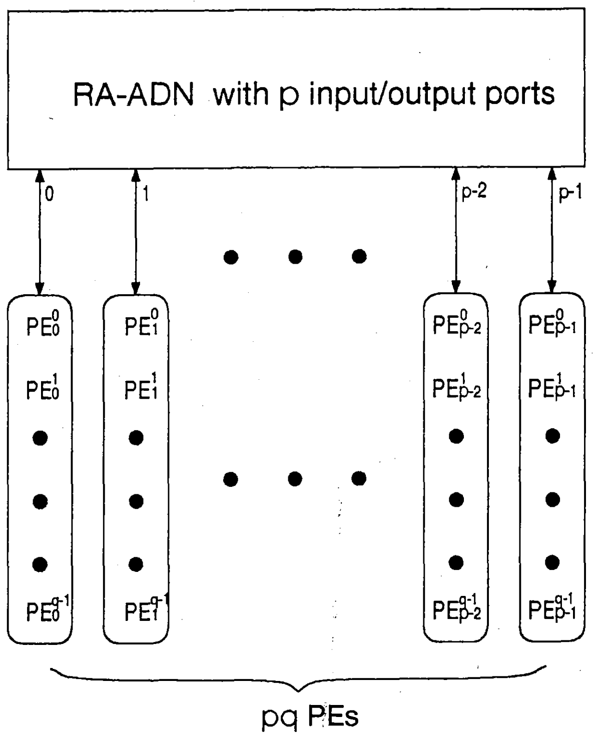

Figure 12: Restricted access RA-EDN System

network cycle is the time needed to route a permutation of S_p in the network of size p). As there are q processors (and thus messages) in every cluster, routing f requires at least q network cycles. Furthermore, a schedule is needed to determine which processor in each cluster is to send at every cycle. Although it is be desirable that the schedule guarantees that at every cycle the destinations of the selected processors belong to mutually distinct clusters so that the resulting communication pattern is a permutation executable by the network in a single cycle, this schedule can be very expensive to compute [31] even when the network is capable of performing arbitrary permutations. Thus we assume a random schedule where at every cycle, any processor whose message is not yet delivered is chosen from each cluster at random. It should be noted that a random schedule on a fixed permutation is equivalent to a fixed schedule on a random permutation. If there are conflicts in the network, then some messages will not be delivered and will have to wait for a subsequent cycle.

At any cycle, at most one processor with an undelivered message is selected per cluster. The selected processor in every cluster i is then put through a q-to-1 multiplexor in the output register O_i, and the destination address of the selected processor is expressed in its 2-digit form (say $\overline{x_i y_i}$). x_i is used as a header and y_i is appended as a trailer. Then, the headers (x_i)'s are used to establish a path between the inputs and outputs of the network of $RA-EDN(b,c,l,q)$. If a path between an input and output cannot be established due to conflicts in the network, then the affected processor is deselected. This processor does not participate further in the network cycle. The trailer y_i in cluster x_i is now used to select the local processor of global label $x_i y_i$ through a 1-to-q multiplexor. At this point a path exists between certain selected processors and their destinations, and messages can be forwarded to their final destinations. These cycles are repeated until there are no undelivered messages.

We are interested in the time required to perform a typical permutation (between the pq processors), as opposed to memory bandwidth, waiting time and probability of acceptance. In the remainder of the paper, we will discuss how the RA-EDN system with an augmented delta network would perform in an SIMD environment.

Conflicts only occur through the EDN, and not after an input has arrived at the destination cluster. Thus the trailer y_i need not be considered in the analysis. We have assumed for the purpose of analysis that we are dealing with a random permutation. Thus the headers x_i which are used as the routing tags of the EDN are uniformly although not independently distributed. However the larger q is, the more closely it approximates a uniform and independent distribution. At the first cycle, there is an output from each cluster, and there is a request at each input of the network. Thus $r = 1$, and the probability of acceptance $P_A(1)$ can be worked out by Equation(4). In this case, it is the fraction of inputs that were routed. The inputs that were not routed are still waiting in the respective clusters to be routed again. r remains at 1 at least for q cycles, and very close to 1 until there are on average only one processor per cluster with an undelivered message e. Thus the average time to have less than one processor per cluster with an undelivered message is $q/P_A(1)$.

At this point there are on average $p*(1-P_A(1))$ processors with undelivered messages, and the probability of a request on any given input is $r = (p*(1-P_A(1)))/p = (1-P_A(1))$. Call this first r, r_1. Using this value of r, there are $p*(1-P_A(r_1))(1-P_A(1)) = p*(1-P_A(r_1))r_1$ processors with undelivered messages after the following cycle, giving $r_2 = (1-P_A(r_1))r_1$. This is continued until $(r_j \times p) < 1$ for some j, say J. At this point it can be assumed that all data can be routed in the following cycle.

Thus the expected time to perform the permutation is:

$$\begin{array}{rl} & q/P_A(1) + J \\ \text{where } r_0 & = 1 \\ r_{j+1} & = (1-P_A(r_j))r_j \\ \text{and } J & \text{is the least value of } j+1 \text{ such that } r_{j+1}p < 1 \end{array}$$

For purpose of illustration, suppose that we have a RA-EDN(16,4,2,16) system, ie., a system with a two stage EDN, and 1024 clusters of 16 processors each. In this system $P_A(1) = .544$. Solving the recursion above gives a J of 5. Thus the expected time to route an average permutation will be about $16/.544 + 5 = 34.41$ network cycles.

6 Conclusions

The Delta network developed by Patel [21] had a much better performance to cost ratio than the traditional crossbar. However, these networks have a unique path between every input and output, and therefore suffered from internal conflicts which reduced performance. So, even though the performance to cost ratio was much higher than the crossbar, the performance relative to the crossbar fell off rapidly with network size.

The Expanded Delta Network describes a family of networks of which the crossbar and the delta networks are specific cases. Members of the family of EDNs exhibit similar performance to crossbar switches for a given size network, but with a cost approximating that of the delta network. In addition, the performance of these networks in an SIMD and MIMD environment is also discussed.

The router network of the MasPar MP-1 computer with 16K PEs can shown to be logically equivalent to the RA-EDN(16,4,2,16) [6].

References

[1] G.B. Adams III and H.J. Siegel, *The extra stage cube: a fault-tolerant interconnection network for supersystems*, IEEE Transactions on Computers, Vol. C-31, No. 5, pp. 443–454, May 1982.

[2] B.D. Alleyne, David A. Kramer and Isaac D. Scherson, *A bit-Parallel, word-Parallel, massively Parallel Processor for Scientific Computing,* Frontiers of Massively Parallel Processing, pp. 176-185, 1990.

[3] K.E. Batcher, *STARAN Series E*, 1977 International Conference on Parallel Processing, pp. 140–143, August 1977.

[4] *Inside the GP1000* (Cambridge, Massachusetts: BBN Advanced Computers Inc., 1988)

[5] V. E. Benes, *Mathematical theory on connecting networks and telephone traffic,* Academic Press, New York, 1965.

[6] T. Blank, R. Tuck, *Personal Communications*, MasPar Computer Corporation, 1991.

[7] C. Clos, *A study of non-blocking switching networks*, Bell System Technical Journal, Vol. 32, pp. 406–424, 1953.

[8] A. Gottlieb et al., *The NYU Ultracomputer - Designing an MIMD shared-memory parallel computer*, IEEE Transactions on Computers, Vol. c-32, No. 2, pp. 175–189, February 1983.

[9] A. Gottlieb and C. P. Kruskal, *Complexity Results for Permuting Data and Other Computations on Parallel Processors,* Journal of the ACM, Vol. 31, No. 2, pp. 193–209, April 1984.

[10] D. Hillis, *The Connection Machine,* MIT Press, Cambridge, Mass., 1986.

[11] K. Hwang and F.A. Briggs, *Computer Architecture and Parallel Processing*, Chapter 7, Section 7.2.4, McGraw-Hill Press, 1984.

[12] D.J. Kuck at al., *Parallel Supercomputing today and the Cedar approach*, Science, Vol. 231, pp. 967–974, February 1986.

[13] T. Lang and L. Kurisaki, *Nonuniform Traffic Spots (NUTS) in Multistage Interconnection Networks*, Proceedings of the International Conference on Parallel Processing, pp. 191–195, August 1988.

[14] D. K. Lawrie, *Access and Alignment of Data in an Array Processor,* IEEE Transactions on Computers, C-24, pp. 1145, 1155, Dec. 1975.

[15] K. Y. Lee, *A New Benes Network Control Algorithm,* IEEE Transactions on Computers, C-36, pp. 768–772, May 1987.

[16] J. Lenfant, *Parallel Permutations of Data: A Benes Network Control Algorithm for Frequently Used Permutations,* IEEE Transactions on Computers, C-27, pp. 637–647, July 1987.

[17] S. Liew and K. Lu, *Performance Analysis of Asymmetric Packet Switch Modules with Channel Grouping*, Frontiers of Massively Parallel Processing, pp. 668-676, 1990.

[18] MasPar Computer Corporation, *MasPar Parallel Application Language (MPL) Users Guide, Software Version 2.0* , MasPar Computer Corporation, Sunnyvale California, 1991.

[19] D. Nassimi and S. Sahni, *An Optimal Routing Algorithm for Mesh-Connected Parallel Computers,* J. ACM, Vol. 27, No. 1, pp. 6-29, Jan. 1980.

[20] D. Parkinson, D.J. Hunt, K.S. MacQueen, *The ATM DAP 500*, Proceedings of the thirty-third IEEE Computer Society International Conference, pp. 196-199, 1988.

[21] J.H. Patel, *Performance of Processor-Memory Interconnections for Multiprocessors* IEEE Transactions on Computers, Vol. C-30, No. 10, pp. 771–780, October 1981.

[22] G.F. Phister at al., *The IBM Research Parallel Processor Prototype (RP3): introduction and architecture*, 1985 International Conference on Parallel Processing, pp. 764–771,August 1985.

[23] C. S. Raghavendra and V. K. Prasanna Kumar, *Permutations on Illiac IV-Type Networks,* IEEE Transactions on Computers, Vol. C-35, No. 7, pp. 662–669, July 1986.

[24] I.D. Scherson, D.A. Kramer and B.D. Alleyne, *A Fine-Grain bit-Parallel, word-Parallel, massively-Parallel Associative Processor,* International Conference on Parallel Processing, Vol. 1, pp. 541-544, 1990.

[25] I.D. Scherson, D.A. Kramer and B.D. Alleyne, *Bit Parallel Arithmetic in a massively Parallel Associative Processor,* IEEE Transactions on Computers, to appear.

[26] H.J. Siegel and R.J. McMillen, *A multistage cube: a versatile interconnection network*, Computer, Vol. 14, No. 12, pp. 65–76, December 1981.

[27] H.J. Siegel, *Interconnection Networks for Large-Scale Parallel Processing,* Lexington Books, 1985.

[28] T.H. Szymanski and V.C. Hamacher, *On the Permutation Capability of Multistage Interconnection Networks,* IEEE Transactions on Computers, Vol. C-36, No. 7, pp. 810–822, July 1987.

[29] T.H. Szymanski and V.C. Hamacher, *On the Universality of Multipath Multistage Interconnection Networks,* Journal of Parallel and Distributed Computing, Vol. 7, No. 3, pp. 541–569, December 1989.

[30] L. G. Valiant, *A Scheme for Fast Parallel Communication,* SIAM Journal on Computers, Vol. 11, No. 2, pp. 350–361, May 1982.

[31] A. Youssef, B.D. Alleyne, I.D. Scherson, *Permutation Routing in Restricted Access Networks,* IPPS 1992, to appear.