Report Number: WUCS-86-03

1986-01-01

On Designing Interconnection Networks for Multiprocessors

Mark A. Franklin and Sanjay Dhar

This paper considers various physical constraints which influence the design of interconnection networks used in multiprocessor systems. Design expressions are presented for implementing an N log N packet passing interconnection network composed for circuit switched crossbar chip modules. Expressions reflecting chip level and board level pin and area constraints are derived and used to determine the network delay expected at a given clock frequency. Logic and memory delay, signal path delay, clock skew and clock tree delay parameters are defined and used to determine the maximum frequency which can be obtained with a given design. An example 2048x2048 network... **Read complete abstract on page 2.**

On Designing Interconnection Networks for Multiprocessors

Mark A. Franklin and Sanjay Dhar

Complete Abstract:

This paper considers various physical constraints which influence the design of interconnection networks used in multiprocessor systems. Design expressions are presented for implementing an N log N packet passing interconnection network composed for circuit switched crossbar chip modules. Expressions reflecting chip level and board level pin and area constraints are derived and used to determine the network delay expected at a given clock frequency. Logic and memory delay, signal path delay, clock skew and clock tree delay parameters are defined and used to determine the maximum frequency which can be obtained with a given design. An example 2048x2048 network design is considered. This example indicates that using aggressive packaging and MOS technology, a rate of about 32 MHz is achievable. However, this frequency, with this network design, would result in one way delay (ignoring blocking and hot spot delays) of about 1 usecond. A read operation from memory requiring a round trip would thus require more then 2 useconds. This represents more than an order of magnitude slowdown when compared with accessing strictly local memory and appears to be a major problem in the design of network centered multiprocessor architectures.

ON DESIGNING INTERCONNECTION NETWORKS FOR MULTIPROCESSORS

Mark A. Franklin and Sanjay Dhar

WUCS-86-03

January 1986

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

This research has been sponsored in part by funding from NSF Grant DCR-8417709 and ONR Contract N00014-8D-C-0761.

For submission to 1986 International Conference on Parallel Processing.

For
Submission
To

1986 International Conference on Parallel Processing

ON DESIGNING INTERCONNECTION NETWORKS FOR MULTIPROCESSORS

Mark A. Franklin and Sanjay Dhar

Center For Computer Systems Design
Departments of Electrical Engineering and Computer Science
Washington University
St. Louis, Missouri 63130

Abstract:
This paper considers various physical constraints which influence the design of interconnection networks used in multiprocessor systems. Design expressions are presented for implementing an $N \log N$ packet passing interconnection network composed of circuit switched crossbar chip modules. Expressions reflecting chip level and board level pin and area constraints are derived and used to determine the network delay expected at a given clock frequency. Logic and memory delay, signal path delay, clock skew and clock tree delay parameters are defined and used to determine the maximum frequency which can be obtained with a given design. An example 2048x2048 network design is considered. This example indicates that using aggressive packaging and MOS technology, a rate of about 32 Mhz is achieveable. However, this frequency, with this network design, would result in a one way delay (ignoring blocking and hot spot delays) of about 1 μsecond. A read operation from memory requiring a round trip would thus require more than 2 μseconds. This represents more than an order of magnitude slowdown when compared with accessing strictly local memory and appears to be a major problem in the design of network centered multiprocessor architectures.

On Designing Interconnection Networks For Multiprocessors

Mark A. Franklin and Sanjay Dhar

Center For Computer Systems Design
Washington University
St. Louis, Missouri

1. Introduction and Overview

The design of effective multiprocessor systems involves numerous interacting elements ranging from parallel algorithms to programming languages to computer architecture. This paper focuses on the computer architecture question and, in particular, on the design of interconnection networks for use in multiprocessor systems. Due to their potentially critical effect on overall multiprocessor performance, interconnection networks have been widely studied. Various studies have focused on their functional properties (permutations, control algorithms) [1,8,12,16], their complexity and performance [4,14,15,21], and their actual design [6,18].

One way of characterizing interconnection networks relates to the style of multiprocessor system in which they are used. At one end of the spectrum are systems where the interconnection network is the central communications component between the processors (as in a message passing system) or between processors and the main memory (as in a shared memory system). The NYU Ultra computer [9], the related IBM RP3 [17] computer and the BBN Butterfly [2] are examples of this style of design. It is convenient to refer to these systems as NETWORK CENTERED multiprocessors since the network is a central resource which has a major effect on overall system performance. This system style encourages viewing memory access and communications interchange in a uniform manner with costs associated with access and communications being roughly independent of physical location. (We ignore questions of cache and processor local memory here. If most accesses are local, then why have a large and complex interconnection network in the middle of the multiprocessor ?).

* This research has been sponsored in part by funding from NSF Grant DCR-8417709 and ONR Contract N00014-8D-C-0761.

Algorithm development in this environment tends to encourage the use of large data structures which span the memory space and develop processor access patterns which are relatively uniform in nature.

At the other end of the spectrum are systems where the processors are embedded within the network itself with direct communications taking place primarily on a physically local basis. The Intel Cosmic Cube [20], the Blue Chip computer and various tree machines [22,3,10] are examples (not exhaustive) of this style. These systems are PROCESSOR CENTERED in that processor performance is a main determining factor in overall performance. This system style encourages viewing the world as being made up of local processing niches with data exchange between processors becoming more costly as one reaches further from the local niche. Algorithms which can take advantage of memory access and communications locality associated with a given interconnection structure can operate effectively in this environment.

In the middle of these two ends of the multiprocessor spectrum are machines where some (typically small) portion of the processing task is allocated to the interconnection network. The idea here is that since data must pass through the interconnection network (and be delayed) anyway on its way to and from memory, why not do some processing on the way.

This paper considers the design of large (several thousand inputs and outputs) interconnection networks for the case of NETWORK CENTERED multiprocessor systems. The emphasis is on the physical design aspects of the network, and the implications of design constraints on network implementation and performance. A simple modified packet switched N log N multistage interconnection network topology is assumed, with each node being implemented as a crossbar switch having a limited amount of packet buffering at its input.

The next section presents overall network topology and operation. Two design approaches to the crossbar network nodes are discussed. Section 3 considers pin and area limitations at both the chip and board levels. Section 4 presents simple models of overall network delay as a function of clock frequency. Section 5 derives data rate equations for the network in terms of

various design parameters such as logic, memory, and data path delays. Clock skew and distribution factors are included in the model. The model presented can be used to determine the maximum clock rate achievable and hence the expected delay through the network. Section 6 explores an example design of a 2048 input/output switch. The conclusion indicates that, in the current design environment, a large NETWORK CENTERED shared memory multiprocessor is likely to encounter a severe performance penalty when accessing memory on the other side of the switch.

2. The Overall Network

The overall network topology considered is of the Boolean hypercube variety as shown in Figure 1. All switch modules are the same and are designed to be crossbar switches sized so that each one fits entirely on a single chip. Earlier studies have indicated that from an area-time performance viewpoint there is little difference between between using an N log N versus a crossbar network within a chip [6]. Across the network as a whole, however, use of a Boolean hypercube structure is significantly less costly in terms of the total number of chips required [7].

While each switch module (being a crossbar) is nonblocking, the network as a whole is a blocking network. The number of stages in the network is $\log_N N'$ where N' is the size of the overall network and N is the size of the crossbar on a chip (see Table 1 for various definitions). Since the blocking probabilities of the network decrease as the number of stages decrease, it is advantagous to place as large a crossbar as possible in each switch module (i.e. increase N as much as possible). This is shown in Figure 2 which contains a plot of blocking probability versus number of stages for a network of size 4096 (based on the formula dervied in 15). Note that reducing the number of stages from 5 to 3 decreases the blocking probability by about 10%.

Note also that the length of off-chip signal lines generally increases as the number of stages increases. Thus, for example, the maximum line lengths between the first and second stages in the network of Figure 1 are shorter than those between the second and third stages.

This is important because of the delays associated with driving these lines. Thus, in general, reducing the number of stages reduces the off-chip delays.

Overall network throughput can be increased by using a modified packet switched rather than a pure circuit switched design. A modified packet switched design places a limited number of packet buffers at the input to each of the switch modules. However, within the module circuit switching occurs. Thus, a packet holds an entire path within each switch module as it passes through that module. On leaving the module the path is released for use by some other packet. Increases in throughput which result from a packet switching approach have been discussed elsewhere [4]. These studies also indicate that most of the potential gain from buffering is achieved with a limited number of buffers (about 4) on each switch module input. For simplicity, in our discussion here we assume a single packet buffer, however, this is not critical to the design analysis which follows. We also assume fixed size packets of length 100 bits. This

VARIABLE NAME	TYPICAL VALUE	DEFINITION
N'	2048	Size of Overall Interconnection Network
N	16x16	Size of Crossbar Switch Module (NxN)
N_p	250	Number of Pins on a Switch Module Chip
N_k	40	Number of Pins on a Switch Module Devoted to Control, Clock and Power Functions
W	1,2,4,8	Width or Number of Lines in a Data Path
P	100	Packet Size in Bits
F	20,40,60,80	Clock Frequency (MHz)
T	2	Time To Send Packet Through Network (μsec)
M_{su}	4,2,1	Number of Clock Cycles to Perform Chip Network Setup Functions (DMUX/MUX Design)
V_{DD}	5	Supply Voltage (volts)
ΔV_{max}	1	Allowable Variation in Supply Voltages (volts)
Z_0	50	Line Driver Impedance (ohms)
L	5	Chip Pin Inductance (nano henries)

Table 1: Variable Name Definitions

packet size is about what is needed to include sufficient bits for data, memory module address, intra-memory module address, and return processor address.

Figure 3 shows an example path through a three stage network. In addition to the input packet buffer, a single bit buffer has been placed at each crossbar output to allow for a limited pipelining capability. Notice that a dotted line is present in the diagram around the input buffers. This indicates the presence of a pass-through mechanism which permits packets to stream through the switch module without going through a buffer filling process if the down stream switch module it requires has an empty input buffer. Under light loading conditions this will allow packets to pass from one switch module to another without being slowed down by buffer fill times.

The broad design philosophy is to keep the switch as simple as possible since simplicity generally leads to speed. In this spirit, combining networks and network information processing other than routing is omitted. Hot spots [18] and other problems are assumed either to be dealt with by the operating system, or are accepted and result in performance penalities which hopefully are partially overcome by having a fast network. The network is thus of the RISC (Reduce Interconnection System and Complexity) style of design.

2.1. Network Control

The general design methodology employed will utilize clocked as opposed to strict asynchronous control. Other studies have shown that given today's design environment, system sizes and data rates, clocked designs adequately meet most performance requirements [5,[23]. At very high clock rates, multiple clocked approaches will need to be considered, however they are not treated here in detail. We assume that a two phase clock is used and that two pins on each chip are allocated for this purpose.

A complete interconnection network requires control provision for:

- path establishment and data transfer
- detection of a blocked path
- indication of end of transmission

• path clearing

Packets are self routing, moving from crossbar chip to crossbar chip according to address bits in the header portion of the packet. Within each crossbar the entire path is held from chip input to output. Feeding back from each input buffer to the associated output of the preceding chip is a buffer full line. This indicates if the buffer is full and thus the path is blocked at that point. Packets are backed up and held in prior buffers until the buffer full line indicates transfer can proceed. For each NxN crossbar chip there are thus $2N$ buffer full control lines, N of these indicating whether its buffers are full, and N indicating whether the buffers downstream are full. All packets proceed in lock step from from stage to stage in the network.

Given fixed length packets, an on-chip counter can determine when packets have completed transmission. Path clearing will occasionally be necessary when certain error conditions occur. We assume that such clearing operations are constitute a type of network reset and that such a drastic action will be initiated by some master processor for the network as a whole. One pin per chip is allocated for this purpose.

Thus, ignoring power and ground for the moment, $2N$+3 control lines are needed per chip.

2.2. Crossbar Design

There are many ways of designing crossbar switches. In this paper we consider two approaches (see Figures 4a and 4b). The first is referred to as the MESH CONNECTED CROSSBAR (MCC) design [4]. In this design, N^2 two by two crosspoint switches are placed on the chip, with each switch having a packet routing capability and one bit of buffering to allow for limited pipelining. Packet routing is thus completely local, the layout is planar and the distance between adjacent switches small. The design is modular and as technology improves larger on-chip crossbars can be easily implemented by replicating the basic switch crosspoints. The area of the entire crossbar grows as $O(N^2)$ while the time delay grows as $O(N)$.

The second design approach is referred to as the DMUX/MUX CROSSBAR (DMC) design [11]. In this case, after $\log_2 N$ bits have arrived at an input port, an input port controller (IPC)

(i.e. a demultiplexer) determines on which output line to route the arriving message. The IPC also signals an output port controller (OPC) (i.e. multiplexer) which selects between multiple input packets that request the same output port. An NxN DMUX/MUX crossbar would require $O(N^2)$ two input gates and have a time delay of $O(logN)$ gate delays. In addition to gate delays, however, there are the line delays associated with the potentially long on-chip lines between the input and output ports and controllers. This is due to the fact that path topology from input to output represents a complete bipartite graph whose layout area grows as $O(N^4)$. Certain results show that the overall delay with this type of crossbar grows as $O(N^2)$ [14].

3. Pin and Area Constraints

One serious constraint in the physical design of large networks is that imposed by the limited number of signal pins available both at the chip and board levels. In this section these pin limitations are explored assuming a Pin Grid Array (PGA) packaging technology that is aggressive, but currently realizable and board edge connectors of standard high performance design.

3.1. Chip Pin Constraints

Pin usage can be broadly grouped into three categories: data pins ($N_{p.d}$), control pins ($N_{p.c}$), and ground and power pins ($N_{p.g}$). The total number of pins on a chip, N_p, is thus given by:

$$N_p = N_{p.d} + N_{p.c} + N_{p.g} \tag{3.1}$$

Since the size of the subnetwork on a chip is NxN and W is the data path width, the number of data pins is:

$$N_{p.d} = 2WN \tag{3.2}$$

The control information required for setting up paths in the network is obtained as part of the data and hence requires no <extra signal pins. However, for each input or output port, a control signal indicating the state of the buffer (full or not) is required. This necessitates $2N$

pins for the bufferfull signals. In addition, we allocate two pins for a two phase non-overlapping clock and a pin for system reset. Hence:

$$N_{p.c} = 2N + 3 \tag{3.3}$$

Normally, when small circuits are considered, especially those that have a small number of signal pins, the use of a single pin for power and a single pin for ground is sufficient. However, for large chips especially those that have a large number of signals all of which can switch at the same time, it may be necessary to allocate more pins for power and ground in order to maintain ground and power voltage variations to within acceptable limits.

Each signal pin has an associated inductance. When a signal switches between its high and low voltage states, the change in current through this inductance causes a voltage to appear across the pin. As the number of signal pins increases, in the worst case, all of them can switch in the same clock cycle, thus causing a large voltage change of either the ground or the power net. To quantify this effect, an expression for the required number of ground and power pins can be developed (see Appendix) based on pin inductance, L, clock frequency, F, line impedance, Z_0, power supply voltage, V_{DD}, the allowable power and ground line voltage variation, ΔV_{max}, and the N and W parameters previously defined.

$$N_g = \frac{4LFV_{DD}N(W+1)}{\Delta V_{max} Z_0} \tag{3.4}$$

This expression indicates that the number of ground and power pins increases linearly with frequency. Thus, power and ground pin allocation will have to change as the frequency of operation changes.

The expressions 3.2, 3.3 and 3.4 can now be substituted into 3.1 and the overall number of pins, N_g determined. Using the parameter values indicated in Table 1, Table 2 indicates N_p as a function of subnetwork size, frequency and path width. Assuming that the maximum number of pins available on a chip is 240, the portion of the table to the left and top of the heavy lines indicate designs that satisfy the pin constraints.

		N - Crossbar Subnetwork Size				
F	W	16	18	20	22	24
10	1	69	77	85	93	101
	2	101	113	125	137	149
	4	165	185	205	226	246
	8	294	331	367	403	442

		N - Crossbar Subnetwork Size				
F	W	16	18	20	22	24
80	1	73	81	90	99	107
	2	107	120	133	146	159
	4	176	198	219	241	263
	8	315	353	392	431	472

Table 2: The number of pins per chip necessary, N_p, for different values of subnetwork size N, frequency F and data path width W.

3.2. Chip Area Constraints

We now consider constraints on chip area and obtain the size of the largest network that can be implemented in a single chip. As mentioned earlier, maximizing the size of the crossbar subnetwork (residing entirely on a single chip) reduces blocking and hence is desirable. The two designs introduced earlier (i.e. the mesh connected crossbar, MCC, and the demultiplexer/multiplexer crossbar, DMC) are explored. The development follows that of Padmanabhan [14].

For the case of the MCC design the key element in estimating the area lies in estimating the area occupied by a two input, two output switch. These switches can then be connected in a mesh to form the crossbar network. Padmanabhan gives a PLA implementation of this switch with a one bit wide data path. He shows that the implementation would occupy a rectangular area of dimensions approximately 100 lambda by 100 lambda. Assume that the above implementation gives the area of the control logic of a switch with a W bit data path, and the area occupied by the data path must be estimated. The data path consists of W lines traversing the switch from left to right and from top to bottom. In addition, W control lines for each set of

data lines must be routed. Assuming that separation between lines, including area for driving and control buffers, is 10 lambda, the dimensions of the rectangular area occupied by the switch becomes $100 + 20W$. Hence the area of the MCC realization consisting of N^2 switches is:

$$A_{\text{MCC}} = N^2(100 + 20W)^2 \tag{3.5}$$

Next consider the area occupied by the MUX/DMUX realization. The area occupied by such an implementation can be broadly divided into the area occupied by the N 1-to-N demultiplexers and N N-to-1 multiplexers, and the area occupied by the WN^2 wires which must be routed from the multiplexers to the demultiplexers. The routing of the wires from the demultiplexers to the multiplexers will be done according to the routing presented by Wise [24] which results in identical wire lengths (see Figure 1 for a 4 by 4 network example). Let the minimum separation between wires be d and the vertical separation between consecutive wire origins and endings be h. The area occupied by such a routing can be shown to be given by:

$$A_{\text{wire}} = (N-1)^4 \frac{h^2W^2d}{\sqrt{4h^2 - d^2}} \tag{3.6}$$

The minimum area is occupied when $h = d$ which results in:

$$A_{\text{wire}} = (N-1)^4 \frac{(Wd)^2}{\sqrt{3}} \tag{3.7}$$

Next estimate the area occupied by the N demultiplexers and N multiplexers. Assume that the demultiplexers are implemented as a binary tree of 1-to-2 demultiplexers. A 1-to-2 demultiplexer with W data bits occupies a rectangular area of dimensions $30W$ by 24 [14]. A tree realization will have a maximum of $N/2$ demultiplexers in the first stage with $\log_2 N$ stages. Thus, the bounding box of this tree is given by $360WN\log N$. The area occupied by N demultiplexers and N multiplexers (assuming a multiplexer occupies the same area as a demultiplexer) is given by:

$$A_{\text{dmux/mux}} = 360WN^2\log N \tag{3.8}$$

The total area of the MUX/DMUX design is given by:

$$A_{DMC} = (N-1)^4 \frac{(Wd)^2}{\sqrt{3}} + 360WN^2 \log N \qquad (3.9)$$

The area expressions of 3.5 and 3.9 are lower bound estimates. Table 3 gives the maximum size network that can be implemented in a chip satisfying the above area constraints assuming that these estimates are increased by a third (to handle line drivers, etc.). The maximum chip dimensions assumed are 1cm by 1cm with a $\lambda = 1.5\ \mu m$.

Examination of Table 2 shows that the largest network that can be implemented in a chip i satisfying the pin constraints is 22x22 with a 4 bit data path. Table 3 indicates that the area constraints limit the network size to an 18x18 network with the DMC design, or a 25x25 network with the MCC design, both for a 4 bit data path. However, we choose a convenient power of 2 network size of 16x16 with a 4 bit data path as our basic subnetwork to be implemented in a chip, satisfying both pin and chip area constraints. We next investigate the layout of the network at the board level, taking into consideration routing of wires on the board and physical connections between boards.

3.3. Board Area Constraints

Assume that the pin layout on the chip package uses a pin grid array with three rows of pins having a separation between adjacent pins of 100 mils. The size of a package with at least 175 pins is about 2 inches on a side. The use of a 16x16 subnetwork makes a 256x256 network reasonable for implementation on a single board. This network has two stages, each stage

	Sub network size N	
W	MCC	DMC
1	37	34
2	32	25
4	25	18
8	17	13

Table 3: The largest subnetwork that can be implemented on a 1cmx1cm chip.

consisting of 16 chips. If a stage of 16 chips is lined up on a single side along a board edge then the board length will be about 32 inches. The routing of wires between these two stages will determine the width of the board occupied by the 256x256 network.

To estimate the area occupied by the routing assume that the routing strategy is similar to that adopted for the chip level DMC implementation. The routing in this case is identical to the DMC implementation of a 16x16 crossbar network. The parameters h and d must, however, be estimated differently. d still remains the minimum separation between wire at the board level with a typical value of 50 mils. Assume the board has two signal layers. The total number of wires to be routed is $N^2(W+1) = 1280$. Each layer then has to route half the total number of wires, that is, 640 wires. Hence the vertical separation between wires is $h = 32000/640 = 50$ mil, the minimum. The area occupied by the wires is then obtained by evaluating (3.7) as 73 square inches. Thus, the width of the layout of the wires is about 3 inches and the longest path on the board is then no more than $32 + 3 = 35$ inches. This layout will be used in Section 6 in examining the design of a 2048x2048 network.

3.4. Board Pin Constraints

At the board level, consideration must be given to the routing of signal wires from one board to the next. Implementing a 256x256 network with a 4 bit data path requires the routing of 1280 wires on the input and output sides. In the last section it was shown that the layout of the chips required each edge of the board to be about 32 inches long. If the wires were brought out to the edge of the board on two layers, it was shown that the separation between wires would have to be 50 mil. This is about the minimum separation between wires on the board that keeps crosstalk among wire to acceptable levels. Commercially available connectors are able to connect up to 100 lines from one side of a board and are no more than 4 inches long. Thus with connectors using both sides of the board, eight connectors can be used for the entire 1280 lines which can be lined up on one edge of the board. Thus the pin constraints at the board level can be satisfied.

4. Network Delay

In this section expressions are presented for the time to transfer a packet through the network (T). This is a one way network delay time and doesn't include memory access time. A best case model is presented in which a lightly loaded network is assumed with no blocking of packets. Packets, in this situation, can thus stream through the entire network from the input to the output being delayed only by chip module setup and pipeline fill times.

For the case of the MCC design, network delay is composed of two components. The first is the time to fill the pipe of crosspoint switches from input to output of the network. The second is the time it takes to transfer the packet once it has arrived at the end of the network. Thus:

$$T_{MCC} = \textit{pipeline fill time} + \textit{packet transfer time} \tag{4.1}$$

In this design the average number of crosspoint switches per chip that a packet passes through is N. The number of stages the packet traverses is $\lceil \log_N N' \rceil$. Thus, the pipeline fill time is $N\lceil \log_N N' \rceil$. The number of bit times associated with a packet transfer is the packet size (P) divided by the path width (W). Thus, the packet delay time is:

$$T_{MCC} = (N\lceil \log_N N' \rceil + P/W)(1/F_{MCC}) \tag{4.2}$$

In the DMC design associated with routing the packet within each chip is a setup time (M_{su}). This time is dependent on the size of the on-chip crossbar in that at least $\lceil \log_2 N \rceil$ bits must be received by the chip before the path can be established. Given a path width of W and a clock frequency of F_{DMC} the setup time associated with a single chip (or stage) through which the message passes is thus:

$$M_{su} = \lceil \log_2 N/W \rceil (1/F_{DMC}) \tag{4.3}$$

To break up long path delays, this design also assumes that a 1-bit buffer is present at the

output of each chip. This acts as a $\lceil \log_N N' \rceil$ pipeline through the network. As in the MCC case, a packet transfer time is also present to account for the time it takes the packet to leave the network when its first bit starts to leave an output port. The resulting overall time is given by:

$$T_{DMC} = \textit{setup time} + \textit{pipeline fill time} + \textit{packet transfer time} \tag{4.4}$$

$$T_{DMC} = ([M_{ss} + 1] \lceil \log_N N' \rceil + P/W)(1/F_{DMC}) \tag{4.5}$$

These network delay expressions have been evaluated and are presented in Table 2. Notice that the results indicate that, even at fairly high clock frequencies, the one way delay through the network is substantial when compared to typical memory cycle times. For example, in the DMC model operating at 40 MHz with a path width of 2, the one way network delay is 1.48 μseconds. Round trip delay, including 200 nanoseconds for memory access, would be 3.16 μsec. . That is, the remote (through the network) memory access is more than an order of magnitude greater than local memory access. Note also that the delay expressions and associated tables do not indicate whether or not the frequencies suggested are achievable. This is dependent on the logic and path delays, and on clock distribution characteristics (e.g. clock skew). These design parameters are explored in the next section.

5. Clock Frequency and Data Rate Expressions

The clock frequency at which a given system can run is determined by a host of factors ranging from effectiveness of the logic design, to chip layout and signal delay consequences of using a particular packaging technology. The final result of these factors are a set of delay parameters which determine the rate at which data can pass from chip to chip. This maximum data rate corresponds to the maximum clock frequency which can be used without errors occurring.

The data rate can be expressed in a general form as follows:

	CLOCK FREQUENCY (MHZ)				
W	10	20	30	40	80
1	14.8	7.4	4.9	3.7	1.9
2	9.8	4.9	3.3	2.5	1.2
4	7.3	3.7	2.4	1.8	.91
8	6.1	3.1	2.0	1.5	.76

MCC (Mesh Connected Crossbar) MODEL

	CLOCK FREQUENCY (MHZ)				
W	10	20	30	40	80
1	11.5	5.75	3.8	2.88	1.44
2	5.9	2.95	1.9	1.48	.74
4	3.1	1.55	1.03	.78	.39
8	1.9	.95	.63	.48	.24

DMC (Demux/Mux Crossbar) MODEL

Table 2: Time Through Network (μsec)
(P=100, 512$\leq N' \leq$4096, N=16)

$$DR = \frac{1}{max[information\ signal\ delays,\ clock\ signal\ delays]} \tag{5.1}$$

Information signal delays consist of logic and memory delays (D_L), information signal path delays (D_p) and delays due to clock skew (δ). The sum of these delays can be associated with the time for a signal to pass between communicating modules (chips or crosspoint switches). Since, with a clocked design schema, this must occur within a single clock cycle, this places a constraint on minimum clock cycle duration (and thus maximum frequency). Taking the worst case (largest) sum of these delays overall communicating modules leads an overall constraint on data rate and clock frequency. These idea can be expressed more formally in terms of time constraints associated with communicating sequential machines [23]. However, for brevity this is not done here.

In a similar manner, delays associated with propagation of the clock signal form another basic limitation on clock frequency. Two types of clock distribution schemes are considered here. In the first (the STANDARD CLOCK SCHEME), the entire clock tree is viewed as an equipotential surface which must achieve a single final voltage in each half of the clock cycle. That is, the entire clock tree must be charged and discharged during the clock cycle. Given that

such clock trees may present a large capacitive load, and that this load grows as systems become larger, this constraint can be a limiting factor in achieving high frequencies in physically large systems. If τ is the time required to charge or discharge the clock tree, then the data rate for this clocking scheme is constrained to be less than $1/2\tau$. Based on the above, the data rate for the case of a Standard Clocked Scheme is as follows:

$$DR_{sc} = \frac{1}{max[D_L + D_p + \delta, 2\tau]} \tag{5.2}$$

Notice also that a relationship exists between τ and δ. That is, as the clock line increases in length, both τ and δ increase with τ being an upper bound on δ. One model that relates τ and δ was developed by Wann and Franklin [23]. The model assumes simple exponential rise times to and from the power supply voltage, V_{dd}. Variations in material properties can result in variations in rise time which can be expressed in terms of maximum and minimum τ values, $\tau_{\max}$ and $\tau_{\min}$. Variations in processing can result in variations in FET threshold voltages which can also be expressed in terms of maximum and minimum threshold voltage values, V_{Tmax} and V_{Tmin}. The resultant expression is:

$$\delta = \tau_{\min}\, ln(1 - \frac{V_{Tmin}}{V_{dd}}) - \tau_{\max}\, ln(1 - \frac{V_{Tmax}}{V_{dd}}) \tag{5.3}$$

Naturally, different parts of the clock tree may be treated differently (e.g. on-chip clock distribution, off-chip distribution) depending on design and fabrication techniques employed.

When dealing with long clock lines the charge/discharge time constraint can severely limit performance. One way of dealing with this is to treat clock lines as transmission lines and, using the memory properties of the line, place multiple pulses on the line at the same time instant (the MULTIPLE PULSE SCHEME). Naturally appropriate matched loading and driving techniques must be employed to prevent pulse reflections from causing excessive signal deterioration. Assuming that these problems can be overcome, logic, memory and clock skew

delays will then be the limiting performance factors. (Note as frequencies increase further other limiting effects will come into play such as the skin effect, cross talk effects, etc.) The second denominator term in (5.2) will be negligible compared to the first and the data rate can be expressed as:

$$DR_{sc} = \frac{1}{max[D_L + D_p + \delta]} \tag{5.4}$$

6. An Example

6.1. Physical Design

Based on the design constraint and data rate expressions developed in prior sections, we now consider the design of an interconnection network with 2048 inputs and outputs. As stated in section 3, pin constraints on the chip limit the maximum size of the crossbar that can be implemented in a chip. A 16x16 network having path widths of 4 bits appears to be a reasonable size as far as pin constraints are concerned, and also in terms of being an even power of 2.

The implementation of a 256x256 network on a single board was shown to be possible, satisfying area and pin constraints on the board. Larger size networks can then be implemented from these boards by racking the boards in three dimensional space to reduce the distances over which wires must be routed. A 2048 input, 2048 output network implementation is shown Figure 5. The first two stages of the network are implemented from eight 256x256 network boards; the last stage consists of eight boards with each board implementing one eighth of the last stage. If the boards are stacked as shown in Figure 5, the longest wire between any two chips is one that traverses the diagonal of a board. In section 3 we showed that this distance is 35 inches.

6.2. Clock Frequency

Expressions for the clock frequency (data rate) were derived in Section 5 based on the logic and memory delays and signal path delays and skews in clock distribution. Two types of clock distribution scheme were considered and it was shown that in the MPS (Multiple Pulse Scheme) it is possible to virtually eliminate the dependence of the clock period on the clock delay. In this section we estimate the information signal delays and clock delays from a knowledge of the physical layout of the network.

Information signal delays consist of delay in the logic and in the memory of a finite state machine implementaion. Estimates given in [14] indicate that the logic delay would be about 12 nano seconds whereas the memory delay can be restricted to about 2 nano seconds. This results in $D_L = 14$ nsec. Consider the path delay. Since the network is pipelined, only the largest path delay is of significance. This corresponds to the path that has to go off-chip and traverse the longest distance (35 inches) on the board. The off-chip path delay is determined largely by the speed of signals on the board and this is typically 0.15 nsec/inch. Also, the path delay will include the delay to drive the driver. The driver typically should have a output impedance of 50 ohms to match the line. The delay in driving the driver is about 3 nsec. Hence $D_p = 3 + 0.15*35 = 8.3$ nsec.

The clock distribution essentially consists of two parts: one totally internal to a chip, consisting of a tree distributing the clock to the 16x16 network, and the second external to the chips consisting of the clock distribution on the board. Using a H-tree distribution to minimize clock skew for the clock distribution internal to a chip, the clock delay is given by [25]:

$$\tau_{\text{chip}} = (10N^3 - 3)(3 - 2/N)R_0C_0/7 \qquad (6.1)$$

where R_0 and C_0 represent the resistance and capacitance of the last branch of the tree distributing the clock to a switch. In this design (a 16x16 network occupying a 1cmx1cm chip) we have $R_0C_0 = 0.244$ pico sec. and we get $\tau_{\text{chip}} = 4.1$ nano sec. The delay in the clock distribution on the board is determined in a similar manner as the path delay. It consists of the

delay in driving a driver with a 50 ohm output impedance and the delay in driving a line of maximum length of 35 inches. Thus $\tau_{\text{board}} = 8.3nsec.$. Thus, the total clock delay is $\tau = 12.4$ nano sec. An expression for the clock skew is given in Section 5. Assuming a 20% variation over the nominal value in both the clock delay and transistor threshold voltages, the clock skew is obtained as:

$$\delta = 0.8\tau\log(1 - 2/5) - 1.2\tau\log(1 - 3/5) = 0.7\tau = 8.7 \text{ nanosec.} \tag{6.2}$$

We can now estimate the clock frequencies. Note that due to the pipeline design of the network, only the largest delays are significant and both the MCC and DMC designs resulted in equal clock frequencies. Since, $D_L + D_p + \delta > 2\tau$ the clock frequency for both the standard and multiple pulse clock designs is:

$$F = \frac{1}{D_L + D_p + \delta} = 32 \ \ MHz \tag{6.3}$$

7. Summary and Conclusions

This paper has presented design expressions for implementing an N log N packet passing interconnection network composed of circuit switched crossbar chip modules. The expressions considered are derived from various physical constraints and network models. Chip level and board level pin and area constraints are considered and expressions are derived which indicate the sort of delay which can be expected through a network at a given frequency, and the design constraints on achieving that frequency.

An example 2048x2048 network design is considered. This example indicates that using aggressive packaging and MOS technology, a rate of about 32 Mhz is achieveable. However, this frequency, with this network design, would result in a one way delay (ignoring blocking and hot spot delays) of about 1 μsecond. A read operation from memory requiring a round trip would thus require more then 2 μseconds. This represents more then an order of magnitude slowdown when compared with accessing strictly local memory and appears to be a major problem in the design of network centered multiprocessor architectures.

REFERENCES

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

[2] Bolt Beranek and Newman, "Development of a Butterfly Multiprocessor Test Bed," Quarterly Tech. Rpt. No. 1, March 1985.

[3] S. Browning, "The Tree Machine," Ph.D. Thesis, California Inst. of Tech., Pasadena, 1980.

[4] D.M. Dias and J.R. Jump, "Analysis and Simulation of Buffered Delta Networks," IEEE Trans. on Comput., Vol. C-30, No. 4, pp. 331-340, April 1981.

[5] A.L. Fisher and H.T. Kung, "Synchronizing Large VLSI Processor Arrays," IEEE Trans. on Comput. , 1985.

[6] M.A. Franklin, "VLSI Performance Comparison of Banyan and Crossbar Communications Networks," IEEE Trans. on Comput., Vol. C-30, No. 4, pp. 283-291, April 1981.

[7] M.A. Franklin, D.F. Wann and W.J. Thomas, "Pin limitations and partitioning of VLSI interconnection networks," IEEE Trans. on Comput., Vol. C-31, No.11, pp. 1109-1116, Nov. 1982.

[8] L.R. Goke and G.L. Lipovski, "Banyan networks for partitioning multiprocessor systems," Proc. 1st Annu. Symp. Comput. Arch., pp. 21-28, 1973.

[9] A. Gottleib, et.al., "The NYU Ultracomputer - Designing an MIMD Shared Memory Parallel Computer," IEEE Trans. on Comput., pp. 175-189, Feb. 1983

[10] E. Horowitz and A. Zorat, "The Binary Tree as an Interconnection Network: Applications to Multiprocessor Systems and VLSI," IEEE Trans. on Comput., Vol. C-30, No. 4, pp. 247-253, Apr. 1981.

[11] D.J. Kuck, "The Structure of Computers and Computations: Volume 1," Section 6.5 Typical Alignment Networks, John Wiley and Sons, N.Y., 1978

[12] D.H. Lawrie, "Access and Alignment of data in an array processor," IEEE Trans. on Comput., vol. C-24, pp.1145-1155, Dec. 1975.

[13] C.E. Molnar, T.P. Fang and F.U. Rosenberger, "Synthesis of Delay-Insensitive Modules," 1985 Chapel Hill Conf. on VLSI, Computer Science Press, pp. 67-86, March 1985.

[14] K. Padmanabhan, "Multiprocessor Interconnection Networks In A VLSI Environment," M.S. Thesis, Dept. of Elec. Engr., Washington Univ., St. Louis, Mo., Dec. 1981.

[15] J.H. Patel, "Performance of processor-memory interconnection networks for multiprocessors," IEEE Trans. on Comput., Vol. C-30, No. , pp. 771-780, Oct. 1981.

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

[17] G.F. Pfister, et.al., "The IBM Research Parallel Processor Prototype (RP3): Introduction

and Architecture," Proc. 1985 Inter. Conf. on Parallel Processing, pp. 764-771, Aug. 1985.

[18] G.F. Pfister and V.A. Norton, "Hot spot contention and combiningg in multistage interconnection networks," IEEE Trans. on Comput., Vol. C-34, No. 10, pp.943-946, Oct. 1985.

[19] B. Quatember, "Modular crossbar switch for large-scale multiprocessor systems - Structure and implementation," Proc. AFIPS 1981 Nat. Comp. Conf., vol. 50, pp. 125-135, 1981.

[20] C.L. Seitz, "The Cosmic Cube," Comm. ACM, Vol. 28, No.1, pp. 22-33, Jan. 1985.

[21] H.J. Siegel, R.J. McMillen and P.T. Muller, "Computer interconnection structures: Taxonomy, charactertics and examples," Proc. Nat. Comput. Conf., June 1979.

[22] L. Snyder, "Introduction to the Coonfigurable, Highly Parallel Computer," Computer, Vol. 15, No. 1, pp. 47-56, Jan. 1982.

[23] D.F. Wann and M.A. Franklin, "Asynchronous and Clocked Control Structures for VLSI Based Interconnection Networks," IEEE Trans. on Comput., Vol. C-32, No. 3, pp. 283-291, March 1983.

[24] D.S. Wise, "Compact layouts of banyan/FFT networks," VLSI Systems and Computations, H.T. Kung et. al. Eds. Rockville, MD: Comput. Sci. Press, 1981.

[25] S.Y. Kung and R.J. Gal-Ezer, "Synchronous vs. asynchronous computation in VLSI array processors," Proc. SPIE, vol. 341, May 1982.

APPENDIX

This appendix considers the effect of a changing current in inducing voltage across an inductive load. In particular, an expression is obtained for the voltage induced across a pin when a signal switches between its high and low states. From this the number of pins necessary for power and ground are determined.

The voltage across a pin due to the inductive effect is given by $L\frac{di}{dt}$ where $\frac{di}{dt}$ gives the rate of change of current at the pin. We assume that a signal switches between the voltage levels V_{DD} and 0, then the change in current due to $N(W+1)$ output signal pins switching at the same time is given by:

$$\Delta i = N(W+1)\frac{V_{DD}}{Z_0}$$

The maximum time interval over which this change can take place is one half the clock period, that is, $\Delta t = 1/2F$. Also, the number of ground or power pins is given by $N_g/2$. If the maximum induced voltage is denoted by ΔV_{max}, we then have:

$$\Delta V_{\max} = (\frac{2L}{N_g}\frac{\Delta i}{\Delta t}$$

The number of ground and power pins can then be expressed as:

$$N_g = \frac{2L}{\Delta V_{\max}}\frac{\Delta i}{\Delta t} = \frac{4LFV_{DD}N(W+1)}{\Delta V_{\max}Z_0}$$

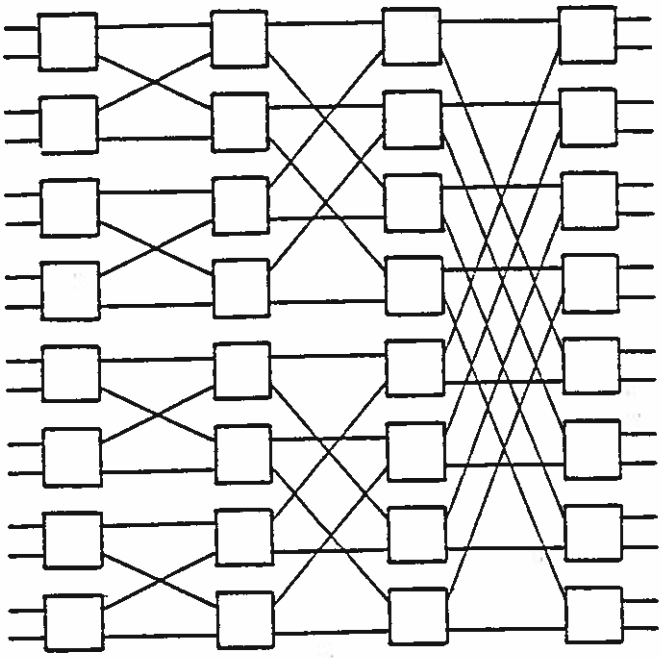

Figure 1: A 16 by 16 N Log N Switch Using 2 by 2 Switch Modules

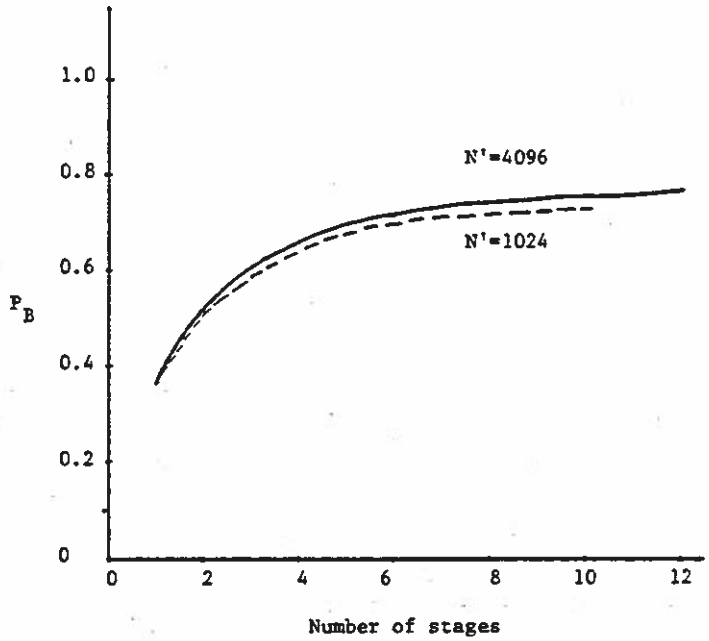

Figure 2: Plot of probability of blocking with the number of stages in a NlogN network. Each stage consists of switches that are crossbar networks.

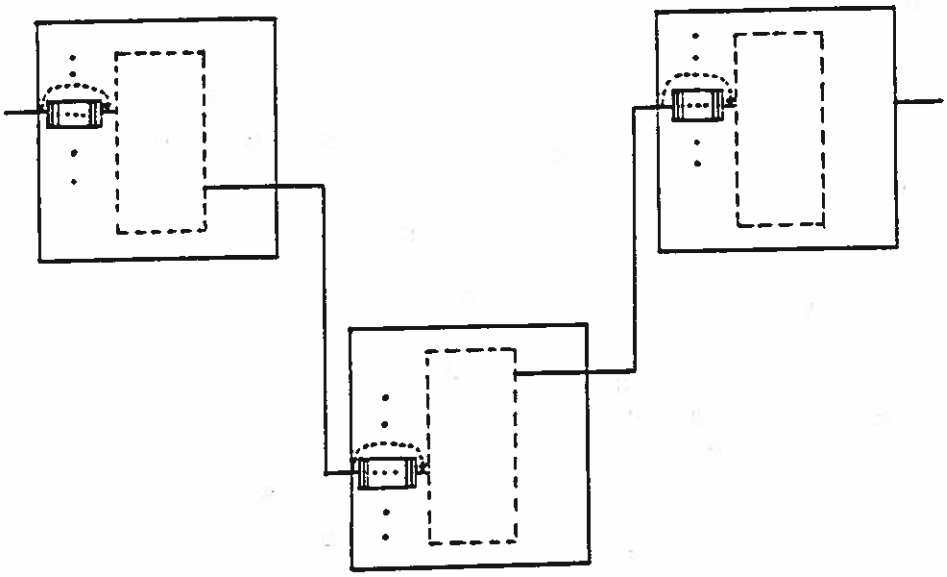

Figure 3: A path from an input to an output in a 3 stage network. Each input at each stage has a buffer which can be bypassed.

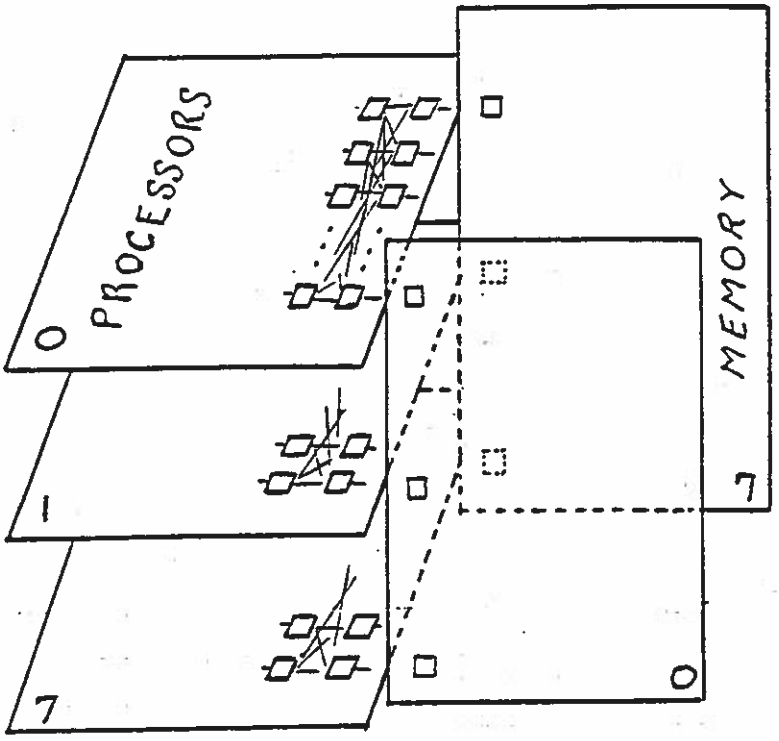

Figure 5: Board Racking Approach to Packaging (ref. 24)

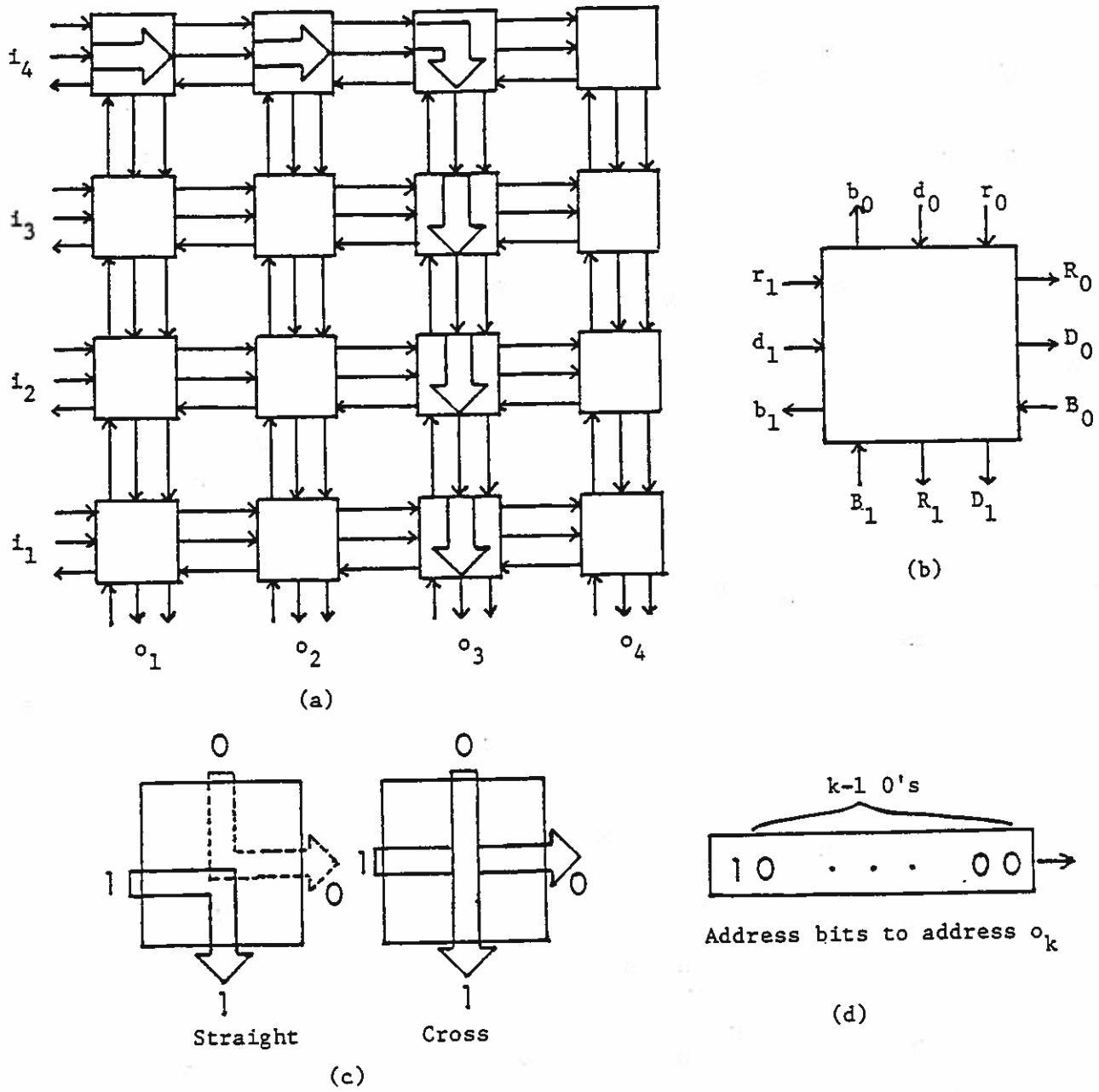

Figure 4a: The mesh connected crossbar network

(a) A 4 x 4 network.
(b) The switching element used.
(c) Possible switch settings.
(d) Address format used.

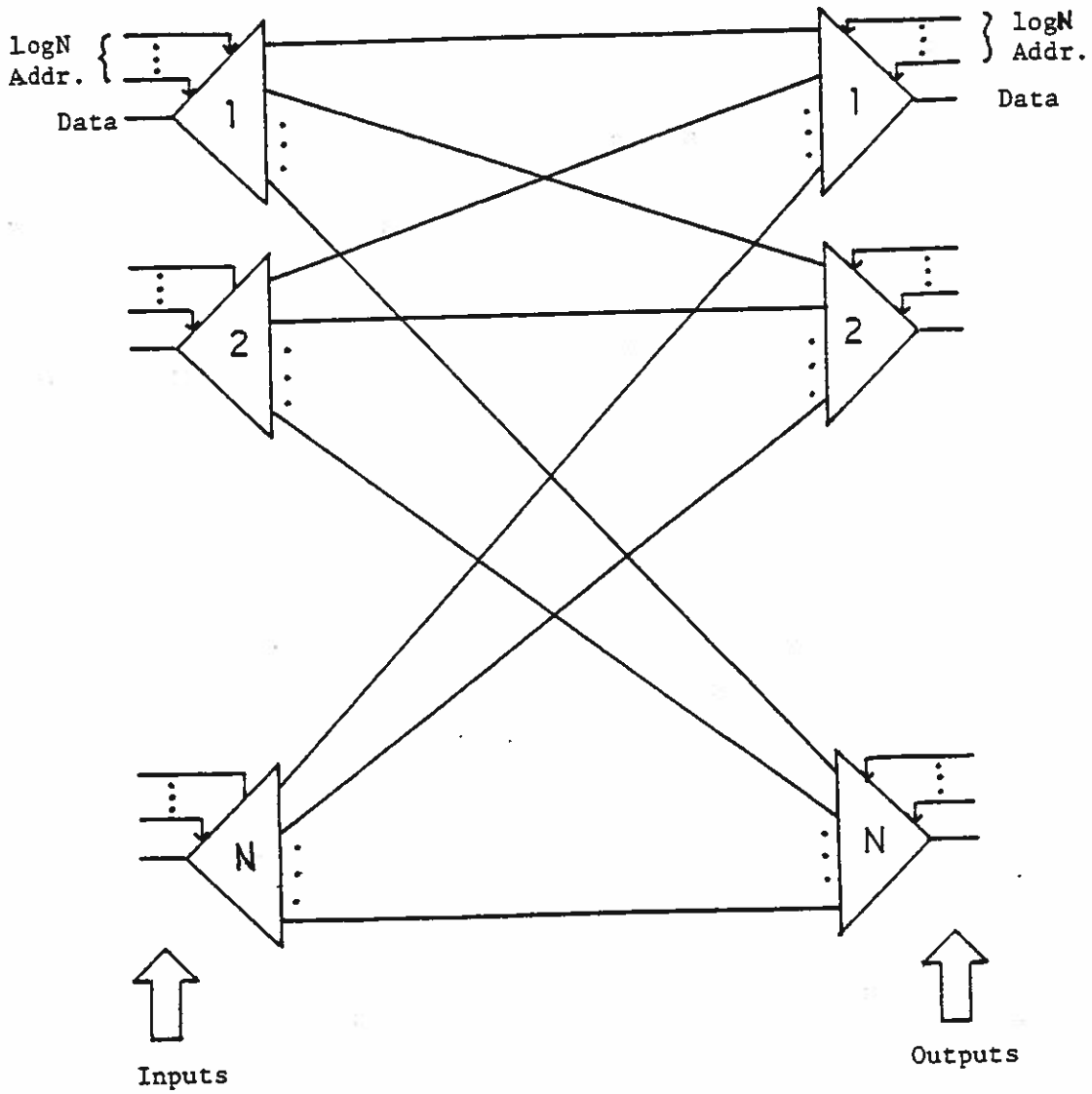

Figure 4b: The MUX/DMUX crossbar network implementation. Each input consists of a 1 to N demultiplexer and each output a N to 1 multiplexer.