High-Speed, Economical Design Implementation of Transit Network Router

by

Kazuhiro Hara

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1995

Author _____

Department of Electrical Engineering and Computer Science
May 12, 1995

Certified by _____

Thomas F. Knight Jr.
Thesis Supervisor

Accepted by _____

Frederic R. Morgenthaler
Chairman, Department Committee on Graduate Students

High-Speed, Economical Design Implementation of Transit Network Router

by

Kazuhiro Hara

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 1995, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

Communication latency is a critical determinant of the amount of exploitable parallelism and the cost of synchronization in massively parallel processors. At the same time, some provisions for fault-tolerance is needed in building a large practical parallel processor system. Because commodity processor technology is improving very rapidly, exploiting this fact through fabrication with the latest technology in the shortest time possible will become more and more desirable to be competitive in the industry. This thesis presents a practical and economical chip design that can be used as a component of a Multistage Interconnection Network called the Transit network for improving both communication latency and fault-tolerance.

Thesis Supervisor: Thomas F. Knight Jr.
Title: Principal Research Scientist

Acknowledgments

I would like to thank Dr. Tom Knight, my thesis advisor, for all his support and help during my thesis. His profound knowledge and insights always amaze me and make me feel that being at MIT is worthwhile in spite of a lot of groaning on my part.

I cannot fully express my gratitude to many of those who helped me out during this process, but I will try. A chief architect of the Transit Project, André DeHon, provided me with the foundations for this thesis and helped me out with CAD tools problems that were time-consuming and grueling experiences for me. His wizardry and knowledge in every aspect of high-performance design and his thick thesis astounded me. Also, Mike Bolotski, Larry Dennison and Rajeevan Amirtharajah gave me a helping hand with Cadence tools. Ed Ouellette tolerated my being his office mate, and tried to cheer me up. I enjoyed his friendliness and good will very much. He also helped me with Mentor Graphics' tools. I am totally indebted to John Leo and Jim Butler, who are not at MIT anymore but helped me keep going through good friendship. Richard and Wendy Hoffman have been great friends of mine and have been good resources in my cultural assimilation. I would also like to thank the floor-mates on the 6th floor, Luis, Rich, Stuart, Matt, Tom and others.

Last but not least, my wife Kazue deserves my sincere gratitude and the degree that I earned. Without her love and understanding, I could not have survived the whole process. She always made me laugh and encouraged me to proceed, contributing to keep my sanity.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

While fiber optics technology provides the necessary bandwidth for transmission of data, the creation of a network that can provide high bandwidth services to the users remains a challenge. One of the difficulties encountered comes from switching. In the telecommunication field, it is projected that such high speed networks will carry various data such as voice, data, images and video in an integrated manner. In such a class of network, provisions to handle wide diversity of data rates and latency requirements are of great necessity. Asynchronous Transfer Mode, or ATM has emerged as a promising technology , which deals with 53 bytes of fixed size packets with line speeds of 155 Mb/s, 620 Mb/s and 2.4 Gb/s.

Whereas data rates and latency requirements are likely to be more uniform in the context of a multiprocessors' communication fabric, no one class of network architecture has yet established itself as such.

As the technology drive continues to enhance commodity processors' performance, it becomes more attractive to build large-scale multiprocessors employing commodity processors from both economic and performance viewpoints. However, the same challenge as the telecommunication field holds here; that is how to provide sufficient inter-processor communication performance at a reasonable cost.

1.1 Problem

Arvind and Iannucci identified memory latency and synchronization overhead as two fundamental issues which limit multiprocessor performance [1]. Communication latency is a critical determinant of the amount of exploitable parallelism and the cost of synchronization

and memory access in massively parallel processors. To hide latency, dataflow architectures [6] and multithreaded architectures [4, 7] have been developed. However, these architectures rely on the abundance of parallelism which fills all the hardware contexts. Even if those latency-hiding schemes are used, it is reported that remote access transactions contribute a large percentage of the cost of running an application [5].

In order to make high-performance scalable multiprocessors, low-latency communication networks are essential. Because communication latency grows with the number of processors in a system, the relative cost of remote transactions increases. This trend indicates that communication latency is a tremendously critical factor.

1.2 Motivations

To build a scalable Multiple Instructions Multiple Data (MIMD) processor, we are concerned about two aspects. In order for MIMD processors to work efficiently, communication fabrics have to offer high bandwidth and low communication latency.

For a system to be truly scalable, fault-tolerant provisions are also necessary, since the number of components and wires in the network increase at least linearly with the number of processors connected in the network. Without such provisions, the overall network performance degrades significantly with a single fault of a switching component.

1.3 Organization

In this thesis, a low-latency network component in such a network and its implementation are presented. In chapter 2, background material is presented for network architectures and component architectures. In chapter 3, the Transit network is described together with a low-overhead protocol and packaging technology. In chapter 4, the chip implementation and design alternatives are discussed in the context of performance and design economics. In chapter 5, verification methodology and testability issues of the chip design are presented. Finally, in chapter 6, conclusions are presented.

Chapter 2

Background

In this chapter, background material is covered for later development of this thesis. We have two questions in mind: What kind of topology facilitates multiprocessor networks to exploit their attributes most? What components enable building efficient networks?

2.1 Network Topology Review

This section describes different classes of switching networks. Before analyzing various network topologies, some terminology needs to be defined. The number of links or channels on a node is called a *node degree.* This reflects the number of I/O ports required per node, thus the cost of a node. Therefore, the node degree should be kept as small as possible. A constant node degree is favorable to achieve modularity for scalable systems.

The *diameter* of a network is the maximum shortest path between any two nodes. To determine the diameter, consider all pairs of nodes and find the shortest path. Then, take the maximum of this path length over all pairs of points. The path length is measured by the number of links traversed. Consequently, the network diameter is a upper bound on network latency; therefore, this should be kept as small as possible to reduce communication latency between nodes.

A network is *symmetric* if it is isomorphic to itself with respect to any node. In a symmetric network, every node sees exactly the same topology.

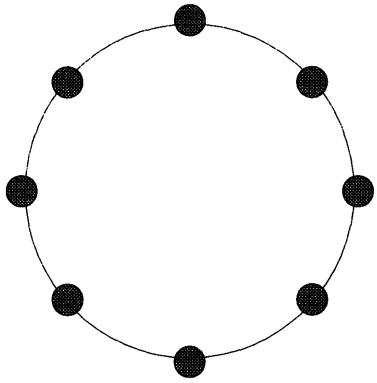


Figure 2-1: Ring

2.1.1 Static Connection Networks (Direct Networks)

Static Connection Networks use direct fixed links once the systems are built. This class of networks is suited to such systems that have predictable communication patterns.

2.1.1.1 Overview

Ring, Chordal Ring and Completely Connected A ring is a two-dimensional network as shown in Figure 2-1, in which N nodes are connected by $N-1$ links in a circle. IBM token ring has this topology, in which messages circulate along the ring until they reach the destination with a matching token.

Chordal rings are obtained by increasing the node degree. In Figure 2-2, chordal rings of degree of three and four are depicted. In the extreme, completely connected network has a node degree of $N-1$ shown in Figure 2-3.

Tree and Star A binary tree has a tree-like structure depicted in Figure 2-4. With a constant node degree, the binary tree is a scalable architecture; however, the diameter is rather long and the root is susceptible to congestion, because cluster-to-cluster traffic has always to go up to the direction of the root.

A star as shown in Figure 2-5 is a two-level tree with a node degree of $N-1$ and a constant diameter of two.

Fat-Tree To alleviate the bottleneck problem, a fat-tree shown in Figure 2-6 is introduced by Leiserson [23]. The channel width of a fat tree increases as we ascend from leaves to the

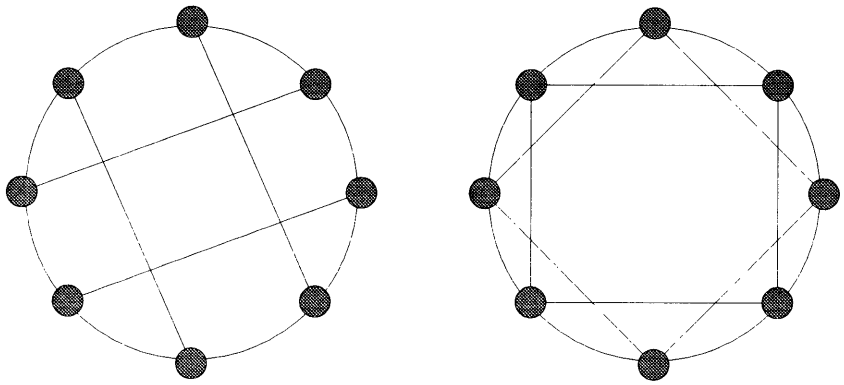


Figure 2-2: Chordal Ring

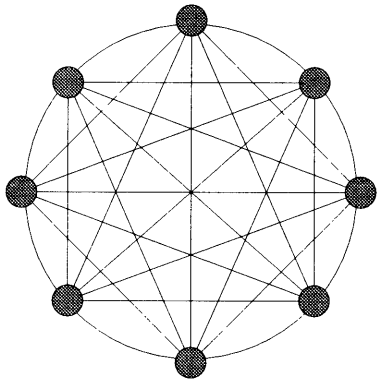


Figure 2-3: Completely Connected

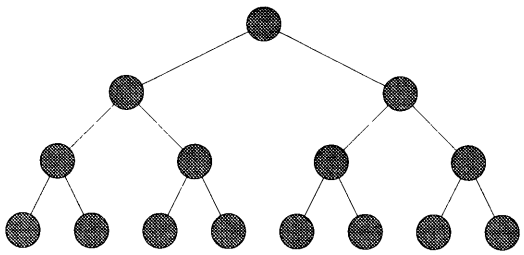


Figure 2-4: Binary Tree

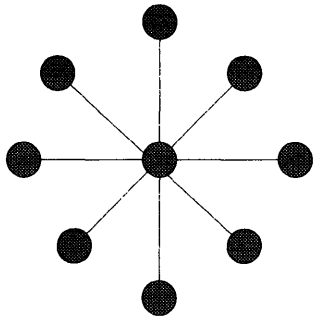


Figure 2-5: Star

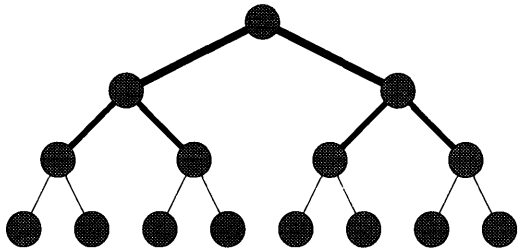


Figure 2-6: Fat-Tree

root. Leiserson shows that with proper channel capacity allocation, fat-trees are *volume-universal networks*, that is, for a given volume of hardware, a fat-tree is nearly the best routing network one can build, and the fat-tree can simulate any other network with at most a polylogarithmic degradation in time. This network architecture is used in Connection Machine CM-5.

Mesh and Torus A mesh network is depicted in Figure 2-7 (1). In general, a k-dimensional mesh with $N = n^k$ nodes has an interior node degree of $2k$ and a network diameter of $k(n-1)$. A torus in Figure 2-7 (2) has ring connections along each row and along each column of the array. In general, $n \times n$ binary torus has a node degree of four and a diameter of $2[\frac{n}{2}]$. Additional wraparound connections reduce the diameter by one-half from that of the mesh network.

Systolic Arrays Static systolic arrays are pipelined with multidirectional flow of data streams as shown in Figure 2-7 (3). With fixed interconnection and synchronous operation, a systolic array matches the communication structure of the algorithm. For special appli-

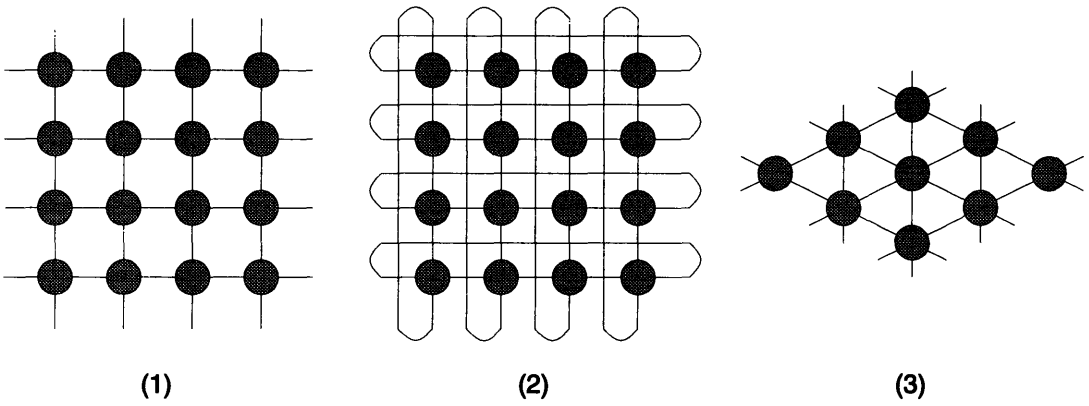


Figure 2-7: (1) Mesh, (2) Torus and (3) Systolic Array

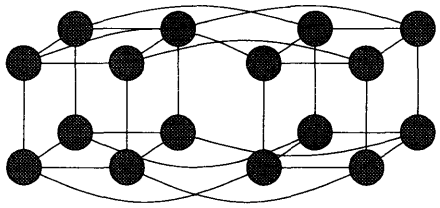


Figure 2-8: 4-cube

cations such as signal processing, this may offer a better cost performance ratio; however, the structure has limited applicability and can be very difficult to program.

Hypercubes (binary n-cube) An n-cube consists of $N = 2^n$ nodes spanning along n dimensions, with two nodes per dimension. An example of $n = 4$ is shown in Figure 2-8. nCUBE and CM-2 were implemented in this architecture. The node degree and the network diameter are both n; therefore, the hypercube is not scalable.

k-ary n-cube In k-ary n-cubes, n represents the dimension of the cube and k is the number of nodes along each dimension. Thus, the number of overall nodes is $N = k^n$. An example of 3-ary 3-cube is shown in Figure 2-9. Rings, meshes, hypercubes and the Omega networks[1] are topologically isomorphic to k-ary n-cube networks.

[1]The Omega networks are described in the later section.

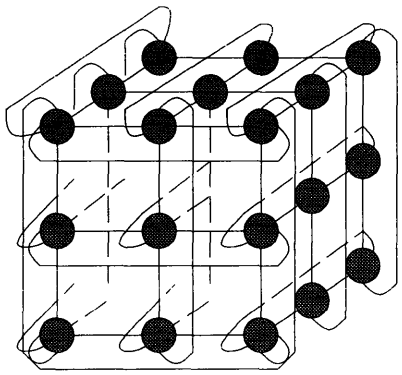


Figure 2-9: 3-ary 3-cube

2.1.1.2 Discussion

Table 2.1 shows some characteristics of static networks. A symmetric attribute of networks has good effects on scalability and routing efficiency. For example, given a uniform traffic pattern, symmetric networks result in uniform loading of network channels. Network cost is affected by the number of links and network diameter. Communication latency is affected by network diameter, but average distance of all nodes is more desirable for this measure. The bisection width is related to the network bandwidth and a lower bound on wire density.

The bottom line for survival of an architecture is packaging efficiency and scalability to allow modular growth. Because of poor scalability and difficulty in packaging higher-dimensional hypercubes, they are being replaced by other architectures. For example, hypercube CM-2 was replaced by fat-tree CM-5. Dally revealed that low-dimensional networks outperform high-dimensional networks when wire density is the limiting factor in constructing the network [13].

2.1.2 Dynamic Connection Networks (Indirect Networks)

For general purpose applications, dynamic connections that can implement any communication patterns are necessary. Instead of using fixed node-to-node links, switches must be used along the paths to provide dynamic connectivity in a Dynamic Connection Network.

Network	Symmetry	Locality	Node Degree	Bisection Width
Ring	Yes	Yes	2	2
Fully Connected	Yes	N/A	$N-1$	$(N/2)^2$
Binary Tree	No	Yes	3	1
Star	No	No	$N-1$	$[N/2]$
Mesh	No	Yes	4	$\sqrt{N}$
Torus	Yes	Yes	4	$2\sqrt{N}$
Hypercube	Yes	Yes	$log_2 N$	$N/2$
k-ary n-cube	Yes	Yes	$2n$	$2k^{n-1}$
MIN	Yes	No	$2k$	$N/2$

Network	Network Diameter	Number of Links	Latency
Ring	$[N/2]$	N	$\Theta(N)$
Fully Connected	1	$N(N-1)/2$	–
Binary Tree	$2(log_2 N-1)$	$N-1$	$\Theta(log_2 N)$
Star	2	$N-1$	–
Mesh	$2(\sqrt{N}-1)$	$2(N-\sqrt{N})$	$\Theta(\sqrt{N})$
Torus	$2[\sqrt{N}/2]$	$2N$	$\Theta(\sqrt{N})$
Hypercube	$log_2 N$	$\frac{N log_2 N}{2}$	$\Theta(log_2 N)$
k-ary n-cube	$n[k/2]$	nN	$\Theta(log_k N)$
MIN	$n+1$	nN	$\Theta(log_k N)$

Comparison is done in the case of N nodes, where $N = k^n$.

Table 2.1: Network Comparison

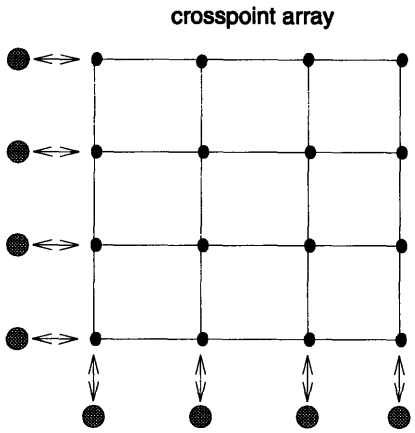


Figure 2-10: 4×4 Crossbar Network

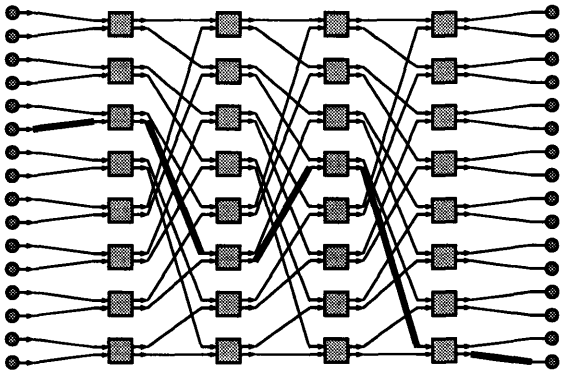
2.1.2.1 Crossbar Network

A crossbar network consists of an array of crosspoints. Each crosspoint switch can provide a dedicated connection path between a source and a destination as shown in Figure 2-10. The switches can be set on or off dynamically according to the demand. Therefore, a crossbar network can implement any permutation without blocking.

2.1.2.2 Multistage Interconnection Networks

Multistage Interconnection Networks (MINs) comprise alternately placed $n \times k$ crossbar switch stage and fixed interstage connections. Different classes of MINs differs in the switch modules and the kind of interstage connection patterns used. Due to a regular structure, MINs are modular, allowing construction of larger networks to use a small network as a building block.

In general, a MIN has $log_k N$ stages, when N is the number of the nodes and k is the radix of the router. Radix is the number of logically distinct directions in which each crossbar switch can route messages. Message routing is controlled by inspecting the destination code of the message header in binary representation. For example, when ith high-order bit of the destination code is a 0, a 2×2 crossbar switch at stage i connects the input to the upper output. Otherwise, the input is directed to the lower output (See Figure 2-11). This way of distributed routing is called *self-routing*, since only an in-band message header is used for routing. A multistage interconnection network is called *non-blocking* if it can perform all possible connections between inputs and outputs.



Thick lines indicate that the message from node 5 to node 15 is routed by the binary representation of the destination address 1111.
(Drawn by André DeHon)

Figure 2-11: 16×16 Omega Network Constructed from 2×2 Crossbars

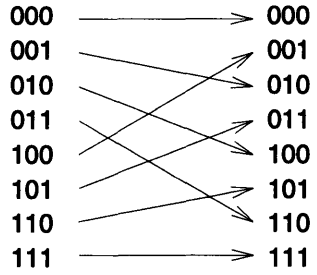


Figure 2-12: Perfect Shuffle

In this section, several MINs are discussed.

Omega Network The Omega network, as shown in Figure 2-11, has the perfect shuffle as interstage connection patterns. Perfect shuffle is a special permutation function for parallel processing application. To shuffle $n = 2^k$ objects, each object is numbered in binary representation such as $(a_{k-1}, a_{k-2}, ..., a_0)$. Then, the perfect shuffle function maps this object to $(a_{k-2}, ..., a_0, a_{k-1})$ by shifting one bit to the left and wrapping around the most significant bit to the right of the least significant bit. An example of this function is illustrated in Figure 2-12.

The Banyan network and the Bidelta network are shown in Figure 2-13 and Figure 2-14 respectively. The Omega network, the Banyan network and the Bidelta network are all blocking networks. In general, MINs are blocking without any provisions that can manage

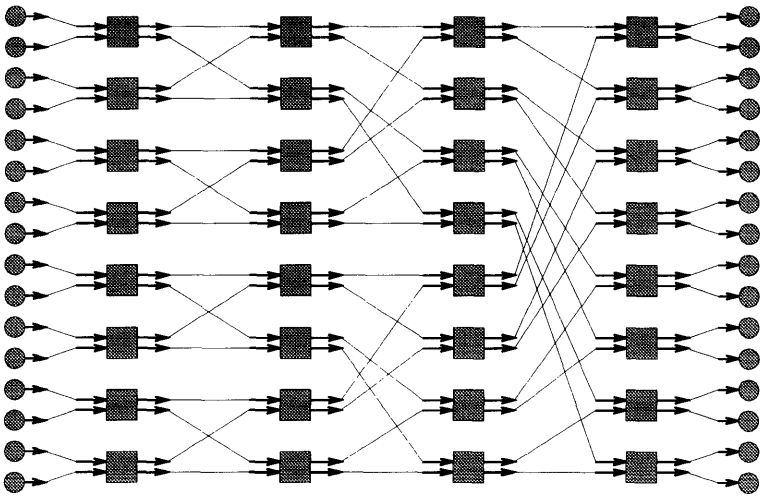


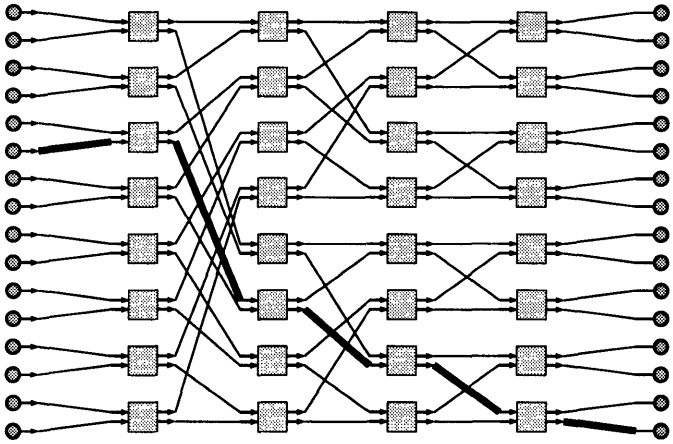
Figure 2-13: 16×16 Banyan Network

output conflicts in the network components such as buffers or redundant paths.

Clos Network One of non-blocking MINs is the Clos network as shown in Figure 2-15. The Clos network is often used in telephone switching systems, since it can be used recursively to construct a larger non-blocking network. Clos network comprises three stages of crossbar switches. The first stage consists of r $n \times m$ crossbar switches, and the second stage consists of m $r \times r$ crossbar switches. The third stage consists of r $m \times n$ crossbar switches. The example in Figure 2-15 illustrates the case where $n = 2$, $m = 3$ and $r = 4$. In general, it is proved that when $m >= 2n - 1$, all permutations are possible [2]. However, the central controller needs to have global knowledge of the states of all switches and to route the traffic appropriately.

Multibutterfly Network A multibutterfly network has a random connection between logically equivalent paths as an interstage connection as shown in Figure 2-16 [28]. A multibutterfly network has a property called *expansion*[2] which, in theory, shows substantial performance and fault-tolerance benefits [28, 25]. This class of networks are further discussed in the following chapter Transit Network.

[2]Expansion is defined in the next chapter Transit Network.



(Drawn by André DeHon)

Figure 2-14: 16×16 Bidelta Network

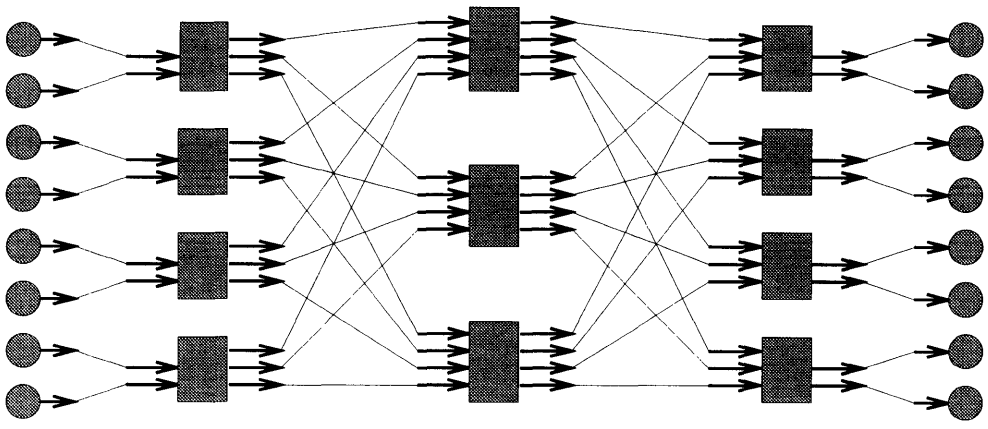
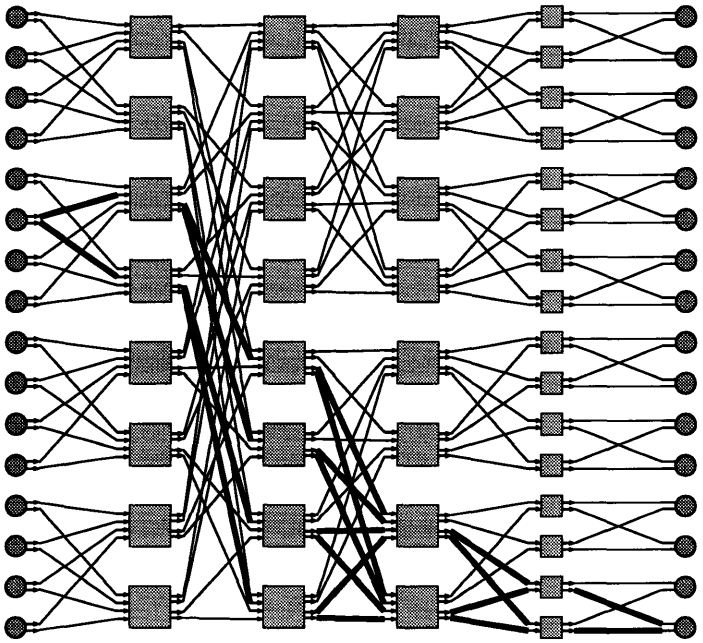


Figure 2-15: Clos Network



(Drawn by André DeHon)

Figure 2-16: 16×16 Multibutterfly Network

2.1.3 Discussion

The highest bandwidth and interconnection capability are provided by crossbar networks. However, the hardware complexity increases in proportion to N^2. If a network is partitioned such that it allows the use of a small crossbar switch as a building block, such an implementation will be the most desirable.

MINs have scalability with modular construction. Moreover, the network latency increases in proportion to $log_{radix}N$ (See Table 2.1 for comparison.). For smaller networks, k-ary n-cubes and MINs are desirable with respect to low communication latency. For fault tolerance and further expandability, MINs are more desirable. Consequently, we focus on MINs in the remainder of this thesis.

2.2 Network Component

This section describes different types of network component architectures.

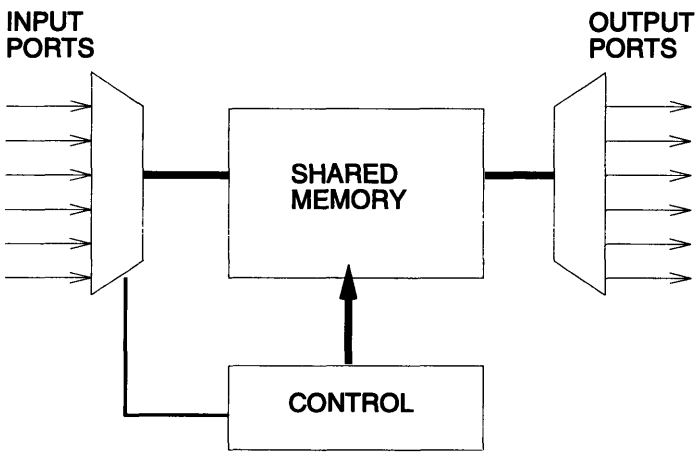


Figure 2-17: Shared Memory Architecture

2.2.1 Shared Memory Architecture

A shared memory architecture switch comprises a dual-ported memory, an input multiplexer and an output demultiplexer as depicted in Figure 2-17. Data arriving on all input lines are multiplexed into a single stream which is written into the shared memory. This memory is organized to provide separate FIFOs for each output line. Simultaneously, the data stored in the shared memory are read out sequentially from each FIFO and demultiplexed to form an output stream for each output line.

2.2.2 Shared Medium Architecture

A shared medium architecture switch multiplexes all incoming data into a single stream onto a common high speed medium such as a bus (See Figure 2-18). Each output port connected to the intermediate bus comprises an address filter (AF) and an output FIFO buffer. The address filter checks the output address of the data on the bus and determines whether the data should be written to the FIFO buffer. Thus, a multiplexed data stream on the bus is demultiplexed into individual streams for each output line. The *ATOM* switch developed by NEC is an example of this architecture.

2.2.3 Space Division Architecture

In a space division architecture switch, point-to-point paths are established from input lines to output lines. There is no multiplexed intermediate entity, so that the maximum data rate

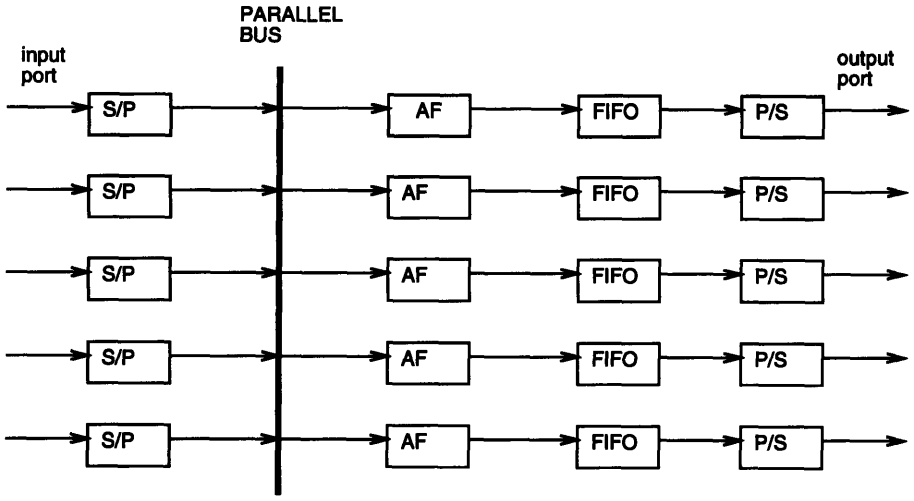


Figure 2-18: Shared Medium Architecture

of the circuit is significantly lower than the other architectures. Moreover, the switching control does not have to be centralized. In the case of Banyan-based configuration as shown in Figure 2-13, each 2×2 crossbar switch can make local decisions, which may lead to less complexity of the circuit. However, space division architecture switches inherently tend to be blocking, depending on implementations of hardware.

2.2.3.1 Crossbar Switch

A crossbar switch consists of an array of crosspoint switches in the same manner as shown in Figure 2-10[3]. In a crossbar switch, as long as there is no output conflict, all incoming data can be routed to respective destinations. Therefore, all permutations are possible.

2.2.3.2 Banyan-based Switches

MIN-based switches have been introduced for the same reason that MIN networks were developed; that is to reduce hardware complexity. N^2 crosspoints must be present in the crossbar network. The switch architecture is constructed in the same configuration as Figure 2-13; therefore, this architecture is inherently blocking without special provisions.

[3]In a crossbar switch, the nodes in the figure are replaced with input ports and output ports.

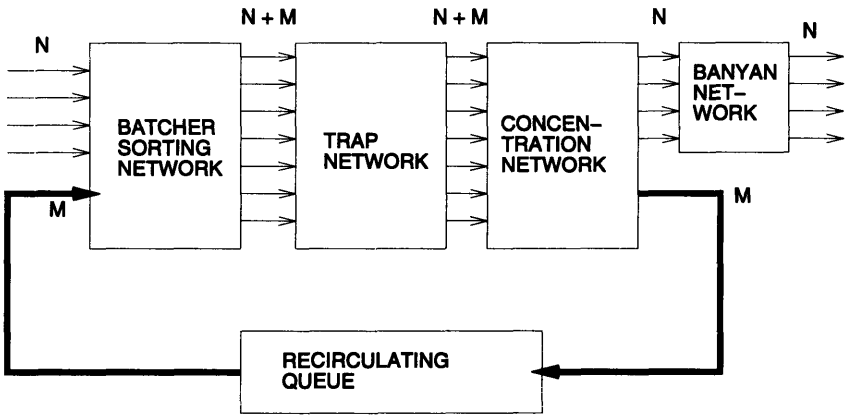


Figure 2-19: Batcher-banyan Switch

Buffered Banyan Switch A simple way to reduce a blocking rate is to put buffers at each input of Banyan switch. Buffered banyan switch has buffers at the inputs to resolve the conflicts. When a conflict occurs, one of the data is chosen to be forwarded and the other remains in its buffer.

Batcher-banyan Switch Another way to reduce a blocking rate is to put a sorting network [3] before Banyan switch. It is proved that any set of packets less than N, which is free of output conflicts, sorted according to destination addresses, and concentrated at the top k lines is realizable by the Omega network [8]. Figure 2-19 shows Batcher-banyan switch architecture. The routing procedure is as follows:

1. The Batcher sorting network sorts the input packets according to their output addresses.

2. Packets causing output conflicts are removed by checking their output addresses over pairs of consecutive output lines of the sorter in the trap network.

3. The remaining packets are concentrated to the top lines by the reverse Omega network.

4. The concentrated packets are routed by the Banyan network.

5. The remaining packets are recirculated and fed into the reserved input ports of the Batcher sorting network.

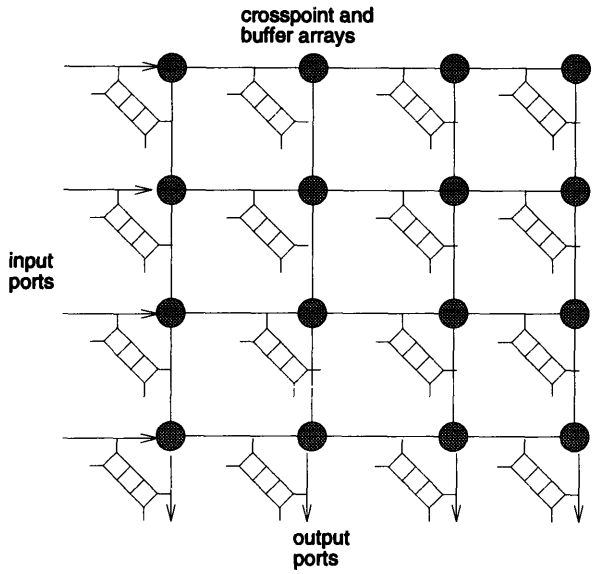


Figure 2-20: Bus Matrix Switch

2.2.3.3 Switches with N^2 Disjoint Paths

Bus Matrix Switch A bus matrix switch has N^2 buffers, one at each crosspoint to resolve output conflicts (See Figure 2-20). Each crosspoint must have switch control logics and FIFO control logics; therefore, this architecture incurs a large amount of hardware complexity.

Knockout Switch Each input port of a knockout switch broadcast data to every output port on broadcast buses as depicted in Figure 2-21. Each output port bus interface has a knockout concentrator (Shown in Figure 2-22 is an example of 8 to 5 concentrator.) which resembles the tournament algorithm. Each pair of two incoming data competes with each other and the winners stay in the same network, whereas the losers go down to another network. (Each tournament network can be viewed as a reversed binary router.) As the same process continues, the necessary subset of data is chosen and multiplexed into an output buffer.

Integrated Switch In an integrated switch, a binary tree is used to route incoming data to the appropriate output port as shown in Figure 2-23. Each output port has a designated buffer for each input port interface to store the data until it is read out. Then the data are

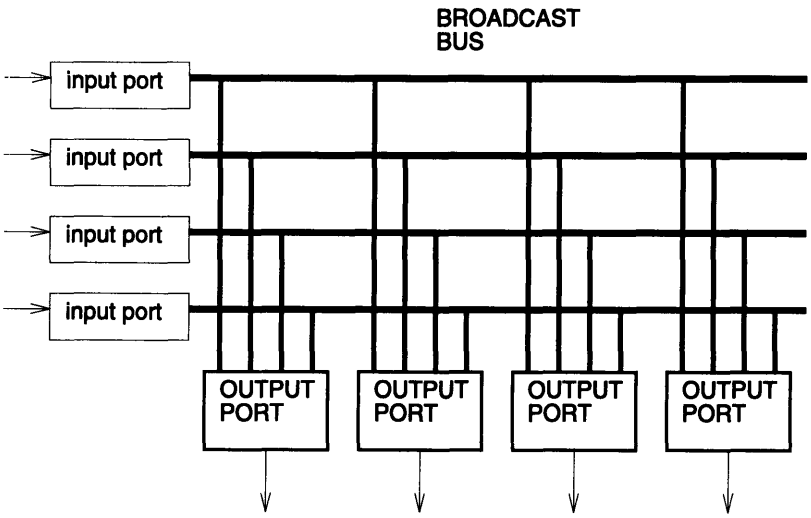


Figure 2-21: Knockout Switch

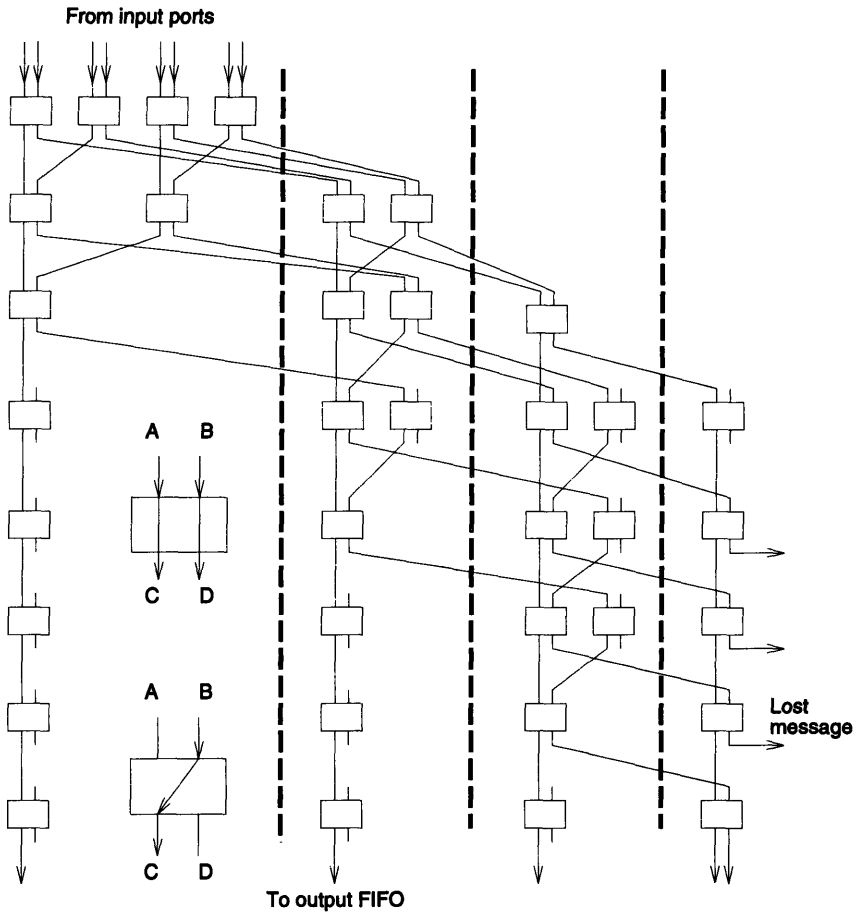
multiplexed into output FIFO.

2.2.4 Discussion

Each component architecture has its advantages and disadvantages. In shared memory architecture switches, memory resources can be shared and yield efficient use of hardware; however, all incoming messages are multiplexed onto the shared memory interface, which is required to operate at N times higher frequency than that of input ports. The central controller which handles all incoming and outgoing messages sequentially is also needed. Those characteristics limit the operating frequency of the circuit implementation.

In shared medium architecture switches, there are distinct FIFOs for each output port so that the operating frequency can be reduced; however, the intermediate bus still needs to operate at N times as high frequency as that of input ports.

In space division architecture switches, multiple concurrent paths from the inputs and the outputs can be established to avoid multiplexing. The controller can be distributed throughout the switches if self-routing schemes are employed. Therefore, the operating frequency of the circuitry is almost the same as the input ports and have the potential to lead to faster switches. Contrary to shared medium architecture switches and shared medium architecture switches, blocking inside the switch can become an issue. In space division architecture switches, it is not possible to have buffers close to the outputs. Instead,



Each box has two states. When there is no message at input A, message B is routed to output C. When there is a message at input A, message A is routed to output C and message B is routed to output D.

Figure 2-22: Knockout Concentrator

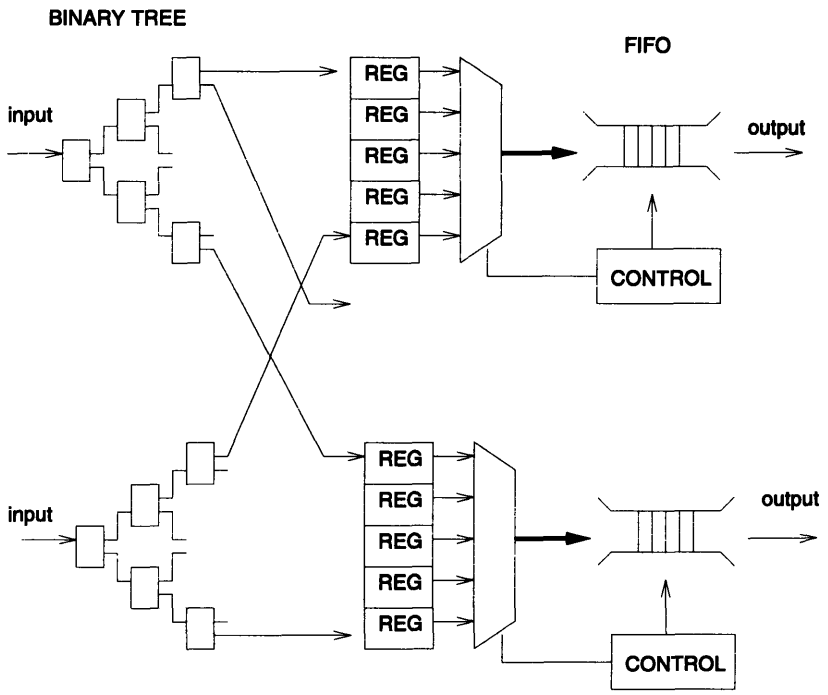


Figure 2-23: Integrated Switch

the buffers must be placed where potential conflicts among paths may occur, or upstream of them.

Buffer placement has an important effect on throughput of the space division switches. Because input buffering mixes messages destined to different outputs in the same queue, it can degrade the performance significantly if the most common first-come-first-served service discipline is employed. While the head of the queue is waiting for the destined output port, the rest of the messages in the queue has to wait even though their destined output ports are available. Instead, checking several consecutive messages in the queue to pick up a message sent to the switching fabric or implementing a central controller which is capable of maximizing an efficient usage of the switching fabric by checking all input queues and scheduling them are needed. However, these schemes incur a great deal of hardware complexity, and are not suited to high frequency operations. Consequently, it is very difficult to achieve the same throughput of output buffering by space division switches with input buffering.

In the multi-processing context, blocking of messages can be tolerable with some provisions of reducing blocking rate and a low overhead protocol. Space division architecture

switches have less hardware complexity, thus have a good potential to exploit high-speed design implementation. Consequently, we opt for space division architecture switches, specifically a dilated crossbar switch[4], and concentrate on it in the remainder of this thesis.

2.3 Summary

In this chapter, various classes of networks and routing components are described. We conclude that the combination of MINs and space division switches, in particular a dilated crossbar switch, is the best choice with respect to low communication latency, fault tolerance and scalability.

[4] A dilated crossbar switch is a crossbar switch which has redundant output ports for each logical output. A dilated crossbar switch is described more in detail in Section 3.2.

Chapter 3

Transit Network

In this chapter, we shall look at a class of networks which focuses on low latency and fault-tolerance in MINs. This class of networks is specifically tuned for multi-processor interconnections in that blocking is assumed tolerable. Because network injection time is small compared with message length, and the source is actually a processor, the penalty of blocking is smaller than for long-haul networks.

Then, a very simple connection-oriented protocol, test provisions and packaging issues are presented. Protocol is an important consideration, since a low overhead protocol substantially increases overall performance. Test interface and packaging are also crucial to augment system performance by fault tolerance and compact efficient designs.

3.1 Characteristics of the Transit Network

There is considerable freedom as to how the multiple paths between the stages in MINs are constructed. In this section, we look at some characteristics which make the Transit network robust and high-performance.

3.1.1 Interwired Networks

To provide fault tolerance in MINs, a *multipath MIN* provides multiple paths between each pair comprised of source and destination. The predominant method has been to construct a network with more stages of switching than are necessary to deliver a message to a destination. Since any of several paths can reach the destination, it is possible to choose a path which avoids any fault in the network. However, the drawback to this scheme is that

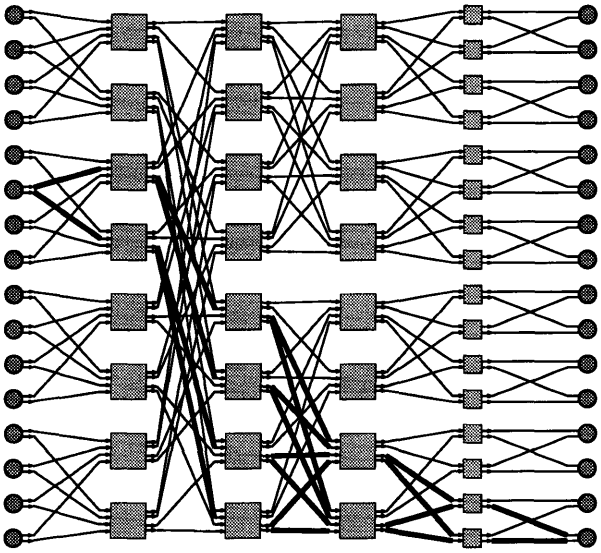


Figure 3-1: 16×16 Transit Network composed of METRO routing components

the communication latency will increase due to the introduction of extra switching stages.

As an alternative to building such networks, *dilated* network components can be used. A dilated network component has more than one output for each logical direction, so that a dilation-d component can have d redundant outputs in each logical direction.

To achieve greater fault tolerance, *interwired* networks can be employed for fixed interstage connections of MINs. Figure 3-1 shows an example of a network constructed with radix-2[1], dilation-2 crossbar switches. As described in the previous chapter, a MIN subdivides at each stage the set of possible destinations into a number of distinct classes determined by the radix of the routing components. Successive stages recursively subdivide the destination class until each output node from the network is uniquely identified. The interstage connection wiring must provide each component in an equivalence class with connections to the appropriate classes. Moreover, in multipath networks, dilated paths must be connected. Interwiring is used to connect these redundant outputs of the components to different components in the appropriate class for fault tolerance. Interwiring ensures that any one component failure does not lead to a loss of connection between a pair of nodes.

[1]Radix is the number of logically distinct directions of the switch.

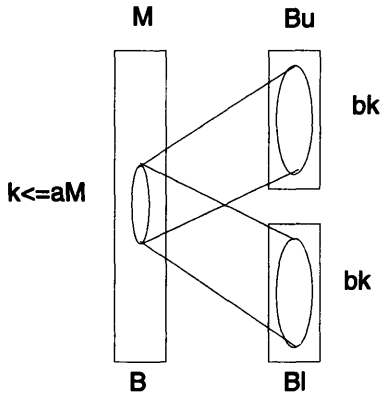


Figure 3-2: An M-input splitter with (α, β)-expansion.

3.1.2 Expansion

In general, the switches in a block B of size $M = \frac{N}{2^s}$ on stage s have neighbors in two blocks, B_u and B_l, on stage $s+1$, where u stands for *upper* and l stands for *lower*. The upper block, B_u, contains the switches on stage $s+1$ that are in the same rows as the upper $\frac{M}{2}$ switches of B. The three blocks, B, B_u and B_l, and the edges between them are collectively called a *splitter*. The switches in B are called the *splitter inputs*, and those in B_u and B_l are called the *splitter outputs*.

Randomized interwirings, with high probability, provide a property of expansion. In particular, an M-input splitter is said to have (α, β)-expansion if every set of $k <= \alpha M$ inputs is connected to at least βk up outputs and βk down outputs, where $\alpha > 0$ and $\beta > 1$ are fixed constants [24]. An M-input splitter is illustrated in Figure 3-2. A splitter network is said to have (α, β)-expansion if all of its splitters have (α, β)-expansion. A splitter network with expansion is called *multibutterfly* network, as described in the previous chapter.

(α, β)-expansion has been proven to have good fault tolerance and performance between groups of nodes.

3.1.3 Maximal-Fanout

Maximal-fanout is an enhancement to fault tolerance by making the paths between any two nodes use as many distinct network components as possible. A network can be said to have *maximal-fanout* up to stage s, if paths between any pair of nodes use $min(d^s, r^{log_r N - s})$ distinct components at stage s. The left term shows that the number of the paths between a

single source destination pair expands further away from the source into the network at the rate of dilation, d. Thus, we can have at most d^s distinct components at stage s. After this stage, the paths will have to diminish by the network radix in order to connect to the proper destination. The number of the rest of the stages is $log_r N - s$; therefore the components at stage s must be less than or equal to $r^{log_r N - s}$. Chong shows that maximal-fanout increases fault tolerance and performance between any single pair of nodes, but sacrifices expansion [10].

Although the worst case bound becomes even worse as the maximal-fanout extends further into the network, Chong also indicates that the lower bound of worst case routing time of permutation such as in bit-reversal and transpose[2] does not look significantly bad, as long as N is not large[3] [10].

3.2 Transit Network

The Transit network is a communication fabric which connects the processing nodes in the MIT Transit Project. The Transit network is an indirect, multistage network built from METRO routing components[4] which are dilated crossbar switches as shown in Figure 3-3; and it is organized with the components in a deterministic, maximal-fanout configuration (See Figure 3-1). A radix-r, dilation-d dilated crossbar switch has $r \times d$ inputs and r logical distinct paths, each of which consists of d redundant outputs.

The goals of the Transit network are low-latency communication, high-bandwidth communication and fault-tolerant communication. Note that communication latency grows as $\Theta(log_r N)$ instead of $\Theta(\sqrt{N})$ in the case of mesh. Consequently, the Transit network can be potentially scalable.

The features of the Transit network are as follows:

- Low latency communication
- High bandwidth, pipelined circuit-switched operation
- Path expansion

[2]By randomizing connections of the interstage network, these worst cases can be avoided. However, it is always possible to generate worst-case permutation.

[3]That is 1024 or less.

[4]The details of the component are described in the next chapter, Implementation.

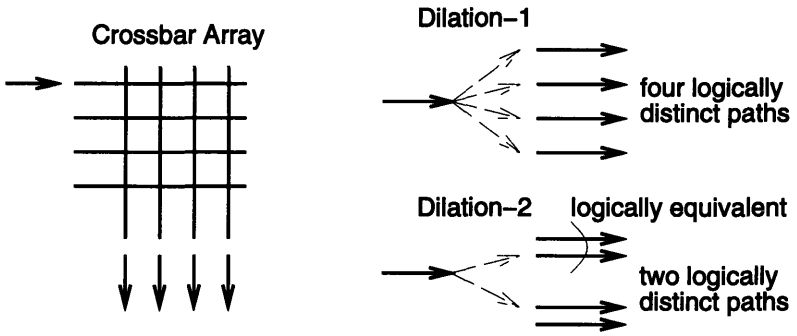


Figure 3-3: Dilated Crossbar Switch

- Fault identification and localization
- Low overhead protocol
- Distributed self-routing
- Gradual performance degradation in the presence of fault

As can be seen in the discussion of maximal-fanout, the first and last stages are most vulnerable to faults. To enhance fault tolerance in the Transit network, every processing node has two links which are connected to different routers. Therefore, it takes at least two router failures to isolate a processing node.

If the system application cannot tolerate any node isolation, clustering nodes and connecting them to the same routers will yield the longest expected time to system failure (See the first stage of Figure 3-1). However, when we lose nodes, all the nodes in the cluster are lost at once.

3.3 Overview of Protocol

The Transit network employs a connection oriented source-responsible routing protocol[5]. A source processor injects a message annotated with a routing header into the Transit network, while retaining a copy of the message. A communication link between a source processor and a destination processor is established semi-permanently, once the routing header reaches the destination. The communication link can be reversed so that an acknowledgment and

[5]This protocol is originally presented in [14].

optional data flow back from the destination to the source. If the connection establishment fails at all, the source processor is responsible to retransmit the message.

The choice to discard blocked messages and retry failed communications leads to a great deal of simplification in the design of routing components, since buffers and precise flow control are not necessary. We do not have to worry about eliminating the duplication of a message caused by faults in the network.[6] In the design of routing components, we opt for simplifications to exploit high-speed circuit technology.

One of the advantages of connection reversal is that an acknowledgment can be combined with data. Hence, actual opening connection traffic does not increase so much. Another advantage is that an acknowledgment cannot be blocked, since the communication link is already established. Note that connection reversal is not mandatory and if so chosen, an acknowledgment can be transmitted as another message depending upon the needs of the applications.

3.3.1 Control Word

Table 3.1 shows the meanings of control words, which command state control of the data transaction in each router through which the message proceeds.

When the connection state is in IDLE[7], a forward port is configured as an input and a backward port as an output whose output is an idle pattern. When ROUTE comes along, the router tries to establish a connection. Depending on the configured dilation mode, the top two (in the case of dilation-1) or one bit (dilation-2) is used to choose one logical output out of four or two. If the connection is established successfully, the control word ROUTE is forwarded to the downstream router[8].

The control word TURN causes the direction of the data transmission established to be reversed. In the forward direction, STATUS and CHECKSUM are returned during the pipeline slacks caused by the connection reversal. The data word STATUS informs the source processor which output port is used. The data word CHECKSUM is generated from the data sent after the connection establishment so that the source processor can verify the

[6] When an acknowledgment, with a less probability than a packet being corrupted, is corrupted, the message can be duplicated in the long run. However, the duplication occurs at a message level, not at a packet level, which is more tractable and less compute-intensive.

[7] The idle state is an initial reset state as depicted in Figure 3-4.

[8] In the case of a dilation-2 Transit network, the interstage wiring should rotate data two bits leftward so that the next router sees the right routing information in the top two bits.

Data Class	Symbolic Representation	Meaning
Control Word	IDLE/DROP	idle pattern or dropping connection
Control Word	ROUTE	direction specification
Control Word	TURN	reversal of connection
Control Word	DATA-IDLE	hold pattern (place holder for connection)
Data	STATUS	connection status
Data	CHECKSUM	checksum bits
Data	DATA	arbitrary data

Table 3.1: Control Word and Data

integrity of the message it sent. In the backward direction, DATA-IDLE is transmitted during the pipeline slacks to hold the connection open until the meaningful data flow in.

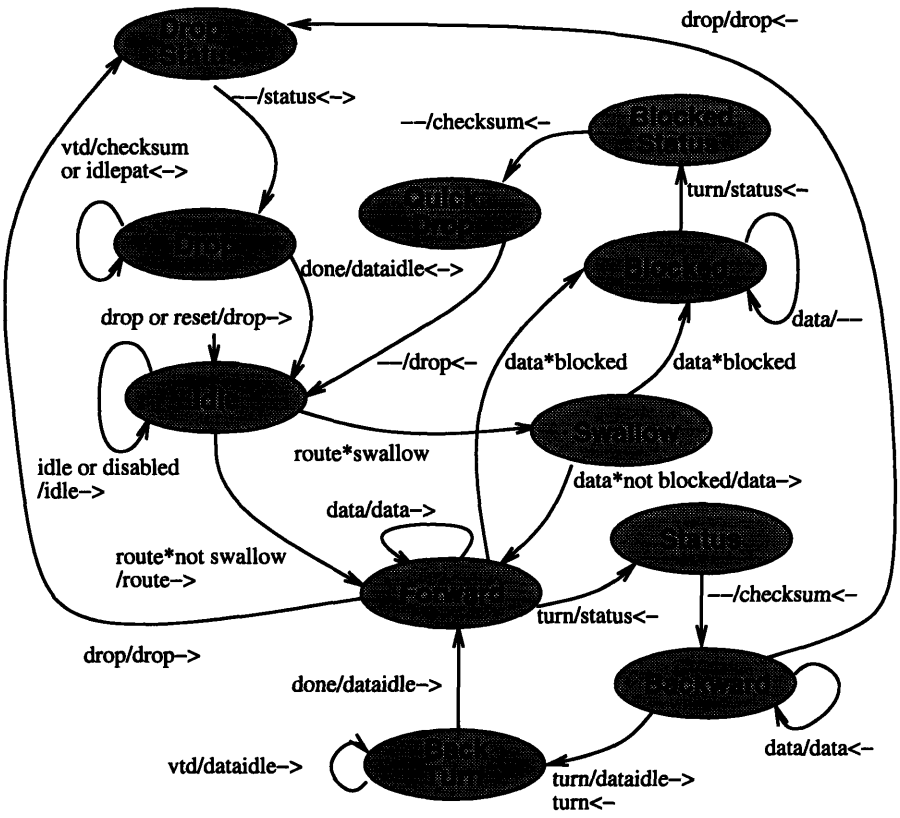
The control word DROP causes the connection to be released, but DROP itself is forwarded before the connection is dropped. Another way that the connection is dropped is by *fast path reclamation.* Instead of waiting for the source processing node to realize that the connection is blocked and to send DROP, a backward control bit (BCB0) is used to notify upstream routers to drop the connection immediately for reuse. This feature helps enhance the network performance and fault tolerance by freeing resources quickly and giving the destination end the freedom to drop the connection in case of faults[9].

3.3.2 Connection State Machine

The connection state transition is depicted in Figure 3-4[10]. The SWALLOW state is only used when there is a defunct control word ROUTE and therefore it has to be removed. As described in the footnote in Section 3.3.1, ROUTE is rotated and consumed partly in each stage. Consequently, when the entire word is consumed and there are more stages to route through, the first ROUTE should be removed and the second ROUTE should be used from the next stage.

[9]There might be a case where the higher-level protocol is terminated, but the data persistently continue to flow in because of a faulty router.

[10]The implemented finite state machines are slightly different from the diagram. The details are discussed in Section 4.2.3.

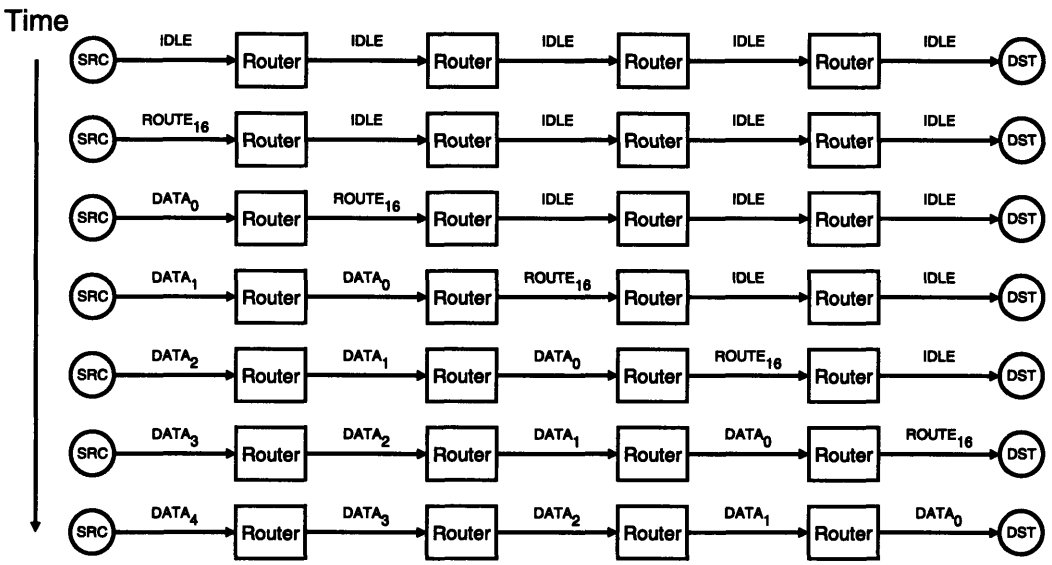


As a connection is opened, used, reversed, and dropped, a link of the network goes through a series of connection states as shown above. In the implementation, the state machine is implemented in a distributed way. (See Section 4.2.3.) Transitions are initiated by the receipt of a control word depending on the local state of the router. Each transition is labeled as (The notation is introduced in [14].): *<event>/<result><dir>*. *<event>* is a control word received or control signals; *<result>* is an output word resulting from the event; *<dir>* is an arrow indicating the direction which the *<result>* is sent.

Figure 3-4: State Transition Diagram

3.3.3 Routing Examples

Figure 3-5, Figure 3-6 and Figure 3-7 show step-by-step examples of state transitions of opening a connection, turning a connection and dropping a connection respectively.



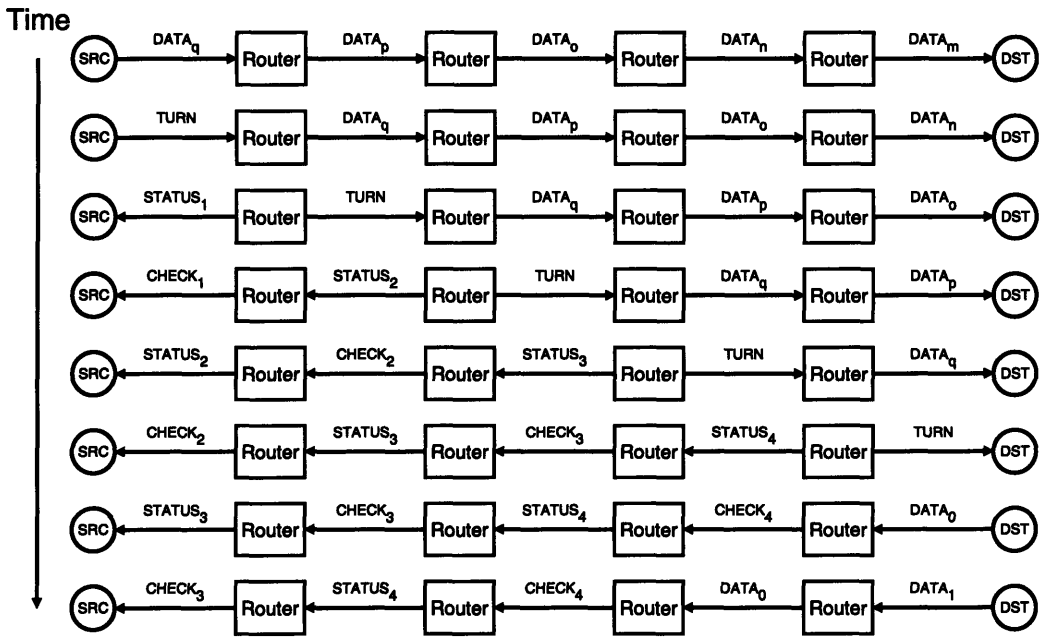
Shown above is a cycle-by-cycle progression of control and data through the network as a connection is successfully opened from one endpoint to another. The message snakes through the network advancing one routing stage on each clock cycle. The first word in the message is the routing word.
(Courtesy of André DeHon)

Figure 3-5: Successful Route through Network

3.4 Scalability

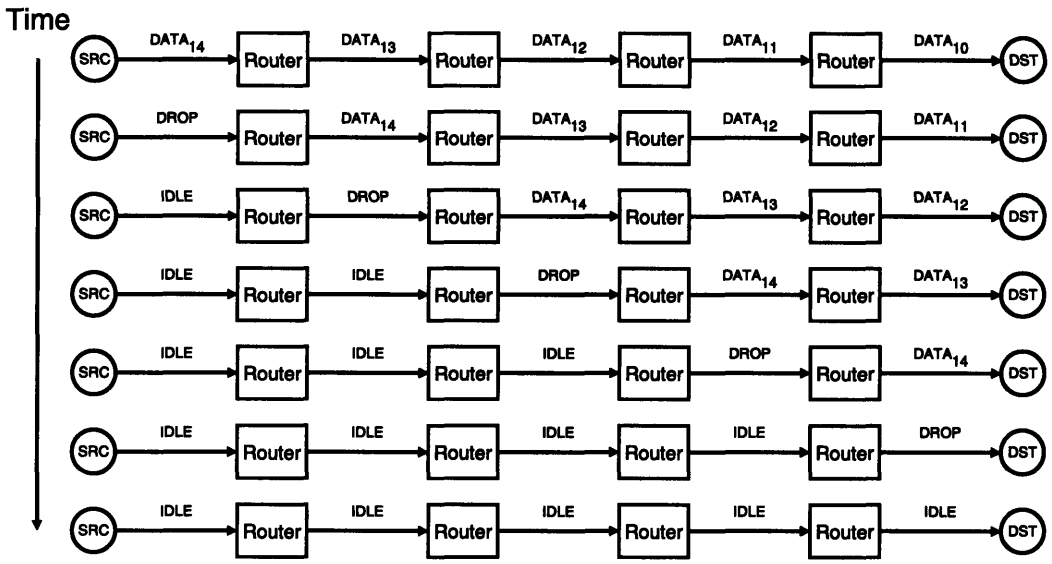
MINs are scalable up to some point, but wire density becomes a problem as the number of processors increases beyond the limit. Also, whatever network topologies are used, the communication latency degrades as the number of the processors is increased. To avoid these problems and to take advantage of communication locality[11], the Transit network can be constructed combined with a fat-tree. Consequently, the average communication latency becomes lower than that of a flat MIN.

[11] All processors do not have to be equally placed/uniformly distributed with respect to distance. Instead, they can be clustered and processes are mapped taking geometric advantage into account. In that case, communications tend to be among the processors in the same cluster.



When the source wishes to know the state of its connection and get a reply from the destination, it feeds a TURN into the network following the end of its forward transmitted data. As the TURN works its way through the network, the links it traverses are reversed. In the pipeline delay required for the link to begin receiving data in the reverse direction, each routing component sends status and checksum information to inform the source of the connection state. After the TURN has propagated all the way through the network and all routers along the connection have sent status and checksum words, data flows backward along the connection.
(Courtesy of André DeHon)

Figure 3-6: Reversing an Open Network Connection



When the transmitting network endpoint decides to terminate a message, it ends the message with a DROP control word. The DROP follows the message through the network resetting each link in the connection to idle after traversing the link.
(Courtesy of André DeHon)

Figure 3-7: Dropping a Network Connection

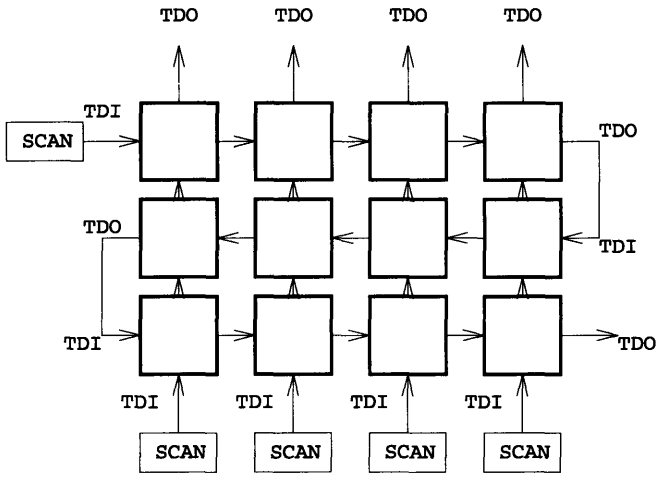
Knight and DeHon introduced a *hybrid fat-tree*, which is a compromise between the close uniform connections in the flat MIN and the locality and scalability of the fat-tree network. In a hybrid fat-tree, the leaves of the hybrid fat-tree are small MINs instead of individual processing nodes [15]. DeHon shows the same dilated crossbar switches that are used in MINs can construct a hybrid fat-tree [14].

3.5 Multiple Test Access Ports

For the purpose of fault tolerance, each routing component in the Transit network has multiple test access ports (TAPs). The IEEE standard test access port and boundary scan architecture is emerging as an industry standard for component and board testing [11]. However, the serial nature of scan based testing is prone to suffer from a single fault in a large-scale system.

Therefore, in the Transit network, multiple TAPs are supported.[12] Any of TAPs can

[12]See an example of dual TAP implementation in Figure 4-8 in Chapter 4.



In a large system, parallel scan is employed. With dual-scan components, only one redundant serial scan will save any of n parallel scan circuitry.

Figure 3-8: Example of Parallel Scan System with Dual-Scan Components.

initiate component testing. When there is a conflict of resources between multiple TAP controllers, the most recently accessing controller has the right of the control and the others are reset to the bypass instruction. Figure 3-8 shows how such a redundant scan circuit is employed in a system. An example of dual TAPs diagram is shown in Figure 4-8 in Section 4.2.7.

To further enhance fault tolerance, a sparse scan instruction encoding scheme which allows strong protection against data corruption was introduced. All legal instruction encodings are sparsely distributed to allow maximun tolerance to erroneous data. All illegal encodings are treated as the bypass instruction so that they cannot interfere with the normal operation of the component. Issues that arise from adapting multiple TAPs are discussed in detail in [14].

3.6 Packaging

The system packaging has an great impact on the system performance. The Transit network is packaged in stack packaging which enables roughly equivalent wiring density in all three dimensions. Figure 3-9 depicts this packaging scheme in which contact between the

printed circuit boards is provided with *button boards*[13] [27]. This contact allows reliable, controlled-impedance contact to be made without solder by using self-aligning compressional connectors.

The routing components are housed in *dual-sided pad-grid arrays* (DSPGAs), which have flat pads located on both the top and bottom of the chip carrier instead of conventional pins [14]. Each vertical pad pair can be connected or disconnected. At the same time, each pad can be connected or disconnected to internal circuitry. Consequently, a pad pair can be used either as a signal path between PCBs, an I/O pad of the same signal or I/O pads of different signals. DSPGA provides through holes for cooling channels.

DSPGAs sandwiched by button boards and PCBs are alternately placed to form a dense three-dimensional stack structure, which is horizontally and vertically aligned so that the signals can travel through this stack structure vertically. Horizontal signals are provided by PCBs. Figure 3-10 depicts an example of the Transit stack structure.

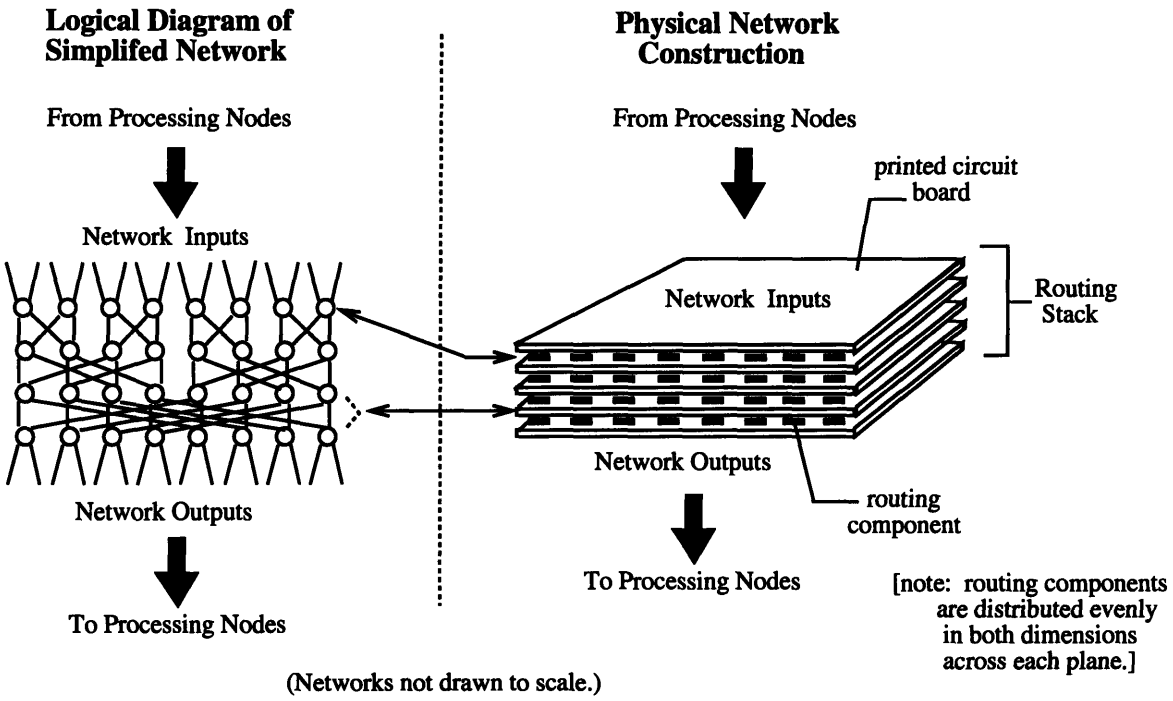
Note that processing nodes can be placed on top of this stack structure[14]. Because the network inputs and outputs of a processing node must be in physical proximity, the return paths from the bottom of the stack must go up straight vertically on additional signaling channels.

The stack packaging can reduce the wire length, so that the communication latency becomes smaller than conventional backplane packaging. The dense packaging does not necessarily mean impractical maintenance and debugging supports. Because there is no soldering, the components are easily repaired and the overall system maintenance, on the contrary, will improve. Debugging packages, whenever necessary, can be inserted to examine the signals between stages, which is also favorable because no permanent means of signal observation[15] is needed in the system.

[13] A button is a compressed fine wire pushed into a cylindrical hole drilled in a printed circuit board. Button boards serve as extremely dense connectors between layers of the packaging.

[14] The processing nodes can also be placed at the bottom, but this incurs mild nonuniformity of communication latency due to the difference of wire length between processing nodes.

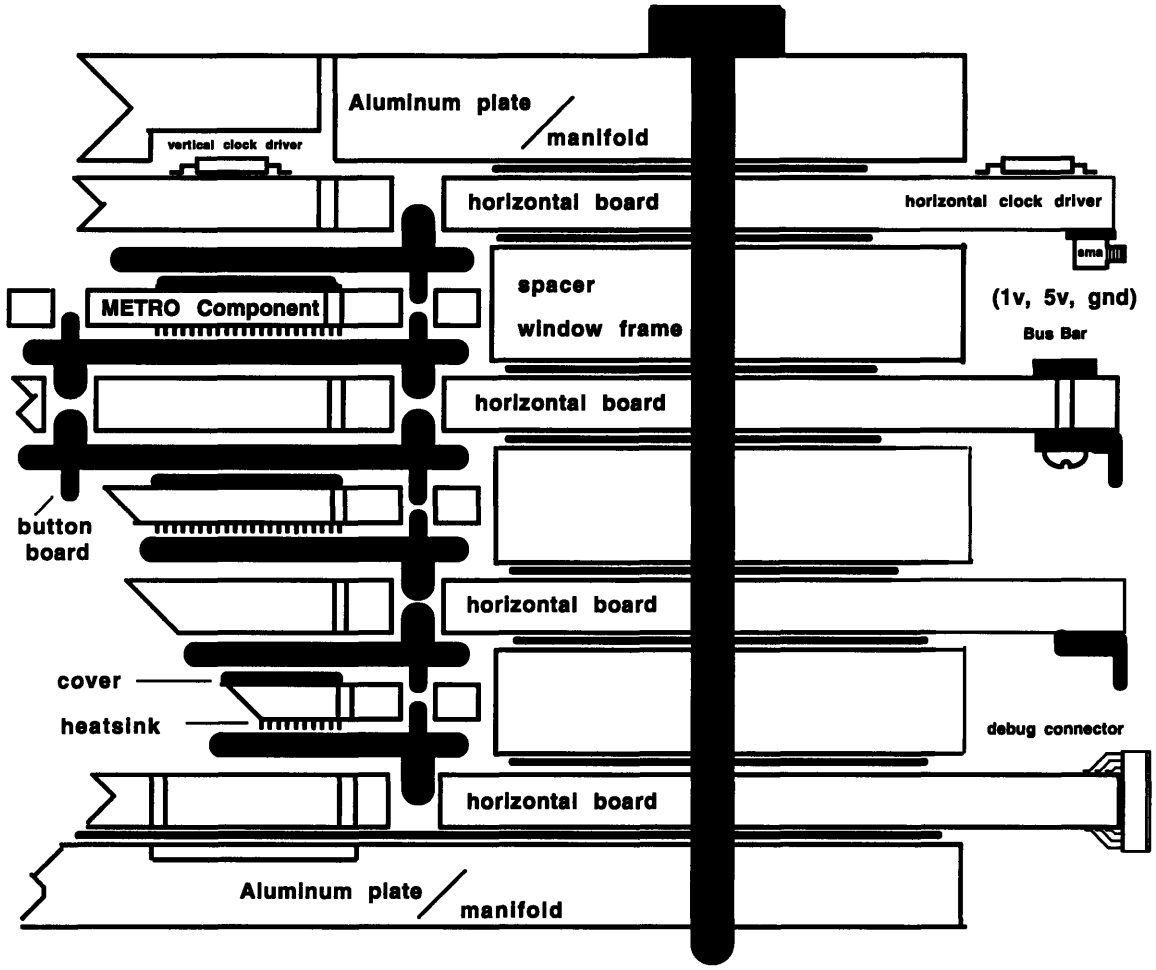
[15] These circuits are usually a burden of system performance.



The diagram above depicts how a logical stack is mapped into the stack structure. The interconnect between each pair of routing stages is implemented as a PCB in the stack. Each stage of routing components becomes a layer of routing components packaged in DSPGA packages.

(Courtesy of André DeHon)

Figure 3-9: Mapping of Network Logical Structure onto Physical Stack Packaging



Shown above is an enlarged cross-section of a network component stack.
(Original Diagram courtesy of Fred Drenckhahn)

Figure 3-10: Cross-section of Routing Stack

Chapter 4

Implementation

As seen in Chapter 2, buffering in the space division switches degrades the communication latency or throughput significantly depending on where the buffers are placed. A scheme using multi-queue input buffers is proposed in [12] to solve the latter problem, but it demands hardware complexity and it is difficult to exploit the technology for low communication latency.

In this chapter, the design implementation of the network component METROSC is presented as a space division switch without buffering in the Transit network described in Chapter 3.

4.1 METRO Routing Component

The METRO routing component, METROSC, is a standard-cell VLSI chip, optimized for low-latency, fault-tolerant communications in tightly-coupled multiprocessors. Complexity is avoided as much as possible in the design, following contemporary trends in processor designs. Rather than supporting message buffering, congestion handling and fault handling in the network hardware, they are passed on to the node processors.

METROSC can be configured as a 4×2 dilation-2 crossbar switch or a 4×4 dilation-1 crossbar switch. When the dilation is configured to one, all the output ports become distinct logical paths. The first two bits of data are used for the routing information that indicates to the allocate block which backward port should be used. When dilation is two, two backward ports are treated as logically equivalent. Consequently, only one bit is consumed as the routing information.

From the standpoint of fault tolerance, the last stage of the network should be composed of dilation-1 components[1], so that the chip is configurable accordingly.

The features of the routing component are as follows:

- Low latency connection establishment and data transmission

  The routing component decodes the routing bits, arbitrates requests, allocates an appropriate backward port and transmits the data in one clock cycle.

- Router width cascading

  To achieve high bandwidth and system flexibility, it is desirable and economical to have capability to cascade chips to widen the data path rather than having a variety of different width routers. This capability can be used to enhance fault tolerance as described in Section 4.3.3.

- Multiple test access ports support

- Port-by-port enabling/disabling

  The input ports and output ports can be disabled port-by-port by way of scan-based TAP. This feature makes it possible to eliminate dead-end paths caused by component or wire failures by disabling corresponding output ports, which results in choosing an alternative path in an upstream router. Input port disabling makes it possible to exclude incoming garbage data caused by wire failures.

- Configurable dilation

  The routing component should be configurable depending on what stage in the network it is located in for the purpose of fault tolerance and system flexibility.

- Connection reversal and variable turn delay

  The connection can be reversed by sending a special command called CONTROL WORD. Since the delay between two stages of the network (that is, the delay between two routing components) may vary, the turn delay is configurable.

[1]Otherwise, we have to connect all equivalent outputs to a destination processing node to deliver a message correctly. This incurs a serious problem in the presence of faulty routers in the last stage, since all the connections to the node are lost by a single point of failure.

- Stochastic path selection

  When the dilation is configured to two, one of the two equivalent backward ports is selected based on a random bit (RND) to reduce the possibility of creating a hot spot in the network. If one port is being used, the other will be selected.

DeHon *et al.* describe the METRO architecture in detail and discusses its projected performance in [31].

4.2 Overview of the Chip Architecture

The chip functional organization and rough floorplan are depicted in Figure 4-1. The crosspoint array is the main source of capacitive load of the datapath. To effectively reduce the transit time, we have to partition the array into forward and backward crosspoint arrays.

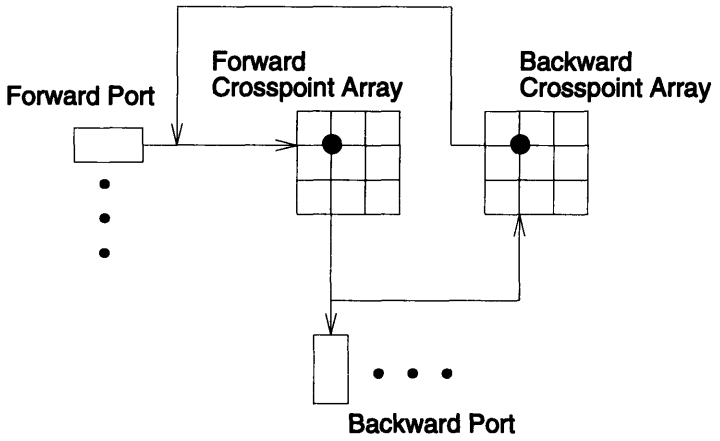
One of the critical paths is in the allocate cycle operation which deals with the opening of a connection from a forward port to a backward port from the results of tentative simulation analysis. In this implementation, regular data flowing through the chip cannot proceed any faster than the time required for the initial allocate operation. Therefore, it is crucial to speed up the allocate path for the entire performance.

The crosspoint array design is also critical to the speed of the data transmission. As all the data go through this array, the propagation delay should be minimal; therefore, the loading capacitance and the length of the paths must be kept to a minimum.

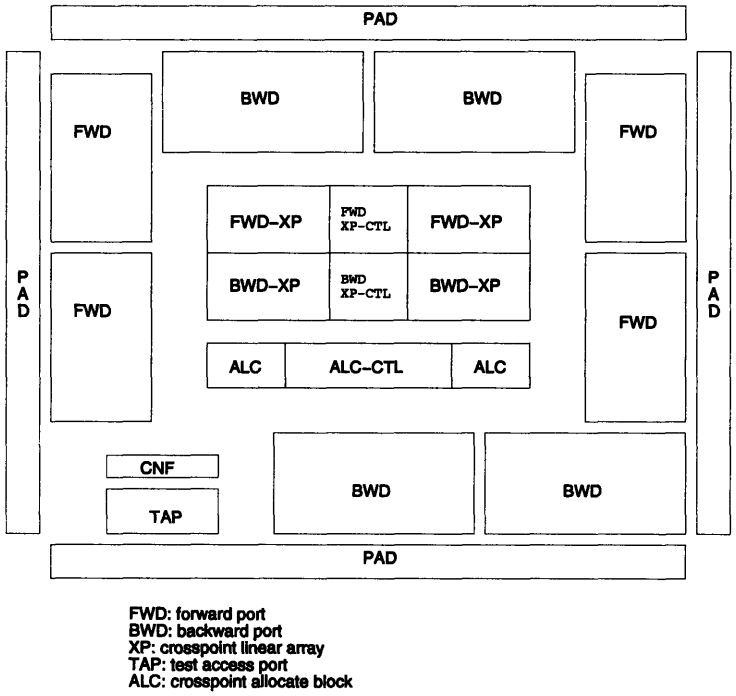
To alleviate these problems, specially tuned allocate cell and crosspoint cell were designed. Most of the designs were done using an optimized standard cell library.

4.2.1 Circuit Switch versus Packet Switch

In long-haul networks such as ISDN, the time it takes to inject an entire message into the network can be small compared to the time required to deliver it; therefore, interconnection bandwidth is utilized most efficiently by packet switching. In contrast, in short-haul networks such as tightly-coupled multiprocessors, the node-to-node communication delays are smaller than or comparable to the message injection and reception time; therefore, if the resource is dedicated to a single message for the duration of its communication, there is little impact on network bandwidth.



(1) All forward ports and backward ports are connected by crossbar switches to provide any combination of connections.



(2) CNF is comprised of configuration register, delay register, impedance register and boundary-scan register.

Figure 4-1: (1) Chip Organization and (2) Chip Floorplan

Circuit switching has some advantages over packet switching in the multiprocessor interconnection context:

- Simple architecture

  Simple architecture makes it possible to employ high-speed circuit implementation.

- No message storage element inside the routing component

  Communication latency can be low. Less complexity yields higher performance and less probability of fault. Buffering control circuits and buffers limit the upper bound of operation frequency. Buffering also causes jitters in transmission time, whereas non-buffering in most cases leads to predictable transmission time. This property is beneficial in multithreaded architecture in which scheduling of multiple threads is necessary.

- Fast, secured acknowledgment

  Connection is already established when acknowledgment starts to come back from the destination.

- Memoryless network

  There is no message stored in the network; therefore, there is no need to duplicate messages in case of fault or to keep track of duplicate messages to prevent them from being delivered to the destination multiple times. Thus, context switching can be done quickly, so that multithreaded architecture benefits a great deal in reducing overhead.

- Preserved message sequence

  All messages are delivered sequentially and corresponding acknowledgment in a handshake manner, so that there is no need to reconstruct messages and check if there are any missing messages or duplicate messages.

A negative aspect is that blocking can occur more frequently. The trade-off between retry penalty due to blocking and buffering delays has to be evaluated. This evaluation typically depends heavily upon network loading and traffic patterns.

Design Style	Max Clock	Chip Size	Power
Full Custom	200%	65%	85%
Standard Cell	100%	100%	100%
Gate Array	75%	120%	110%

Comparison is done by using standard cell design as a basis; therefore, the row of standard cell is always 100%.

Table 4.1: Design Style Comparison

4.2.2 Design Methodology/Economics

In general, we have three options in chip designs; full custom design, standard cell design, and gate array design. Maximum system clock frequency, chip size and power dissipation will be affected by this decision as shown in Table 4.1.

Another consideration is whether static logic or dynamic logic is used in the design. A typical characteristic of precharged dynamic logic is lower transistor count with a reduced capacitive load that gives rise to higher speed capability and less area than static logic. However, dynamic logic sacrifices noise immunity inherent in CMOS technology.

Dynamic logic may generate large current pulses by precharging and evaluating many nodes simultaneously causing dI/dt noise. Precharged dynamic logic cannot recover from noise induced logic errors if the precharged node has been discharged by noise. Static logic, in contrast, may recover unless there is a feedback loop in a circuit.

Unfortunately, the dynamic logic race condition is only apparent in SPICE simulation. Charge redistribution could be a major concern in the circuit designs. The interface of static and dynamic logic must be included in complete simulation with back-annotated capacitances from the layout with effects of power and ground bounce is needed to ensure correct operation.

Static logic, on the other hand, is robust to noise, modular and easy to handle, so that the gates can be easily cascaded to construct complex functions without simulating the circuits extensively for correct operation. A fully static design can accommodate IDDQ (supply current quiescent) testing for finding bridging faults, which improves testability of a chip.

This choice will also affect power dissipation. It is reported that when the logic depth is

larger than a certain value[2], dynamic logic consumes more power than static logic [18]. In our design, messages have to go through the router in one clock cycle; therefore, the logic depth is relatively deep, and consequently favors static logic.

Therefore, design complexity, size, power dissipation, and reduced noise immunity have to be taken into account in deciding a design scheme. In the design of METROSC, we choose the static logic in standard cell design for ease of development. The macro cells' layout style generally complies with the standard cell library so that overall chip layout can be done efficiently and easily.

By using macro cells in the standard cell design to speed up circuits in critical paths, relatively high performance can be obtained without exhaustive simulation and tuning efforts, both of which will arise in the dynamic logic in the full custom design.

4.2.3 Forward Port

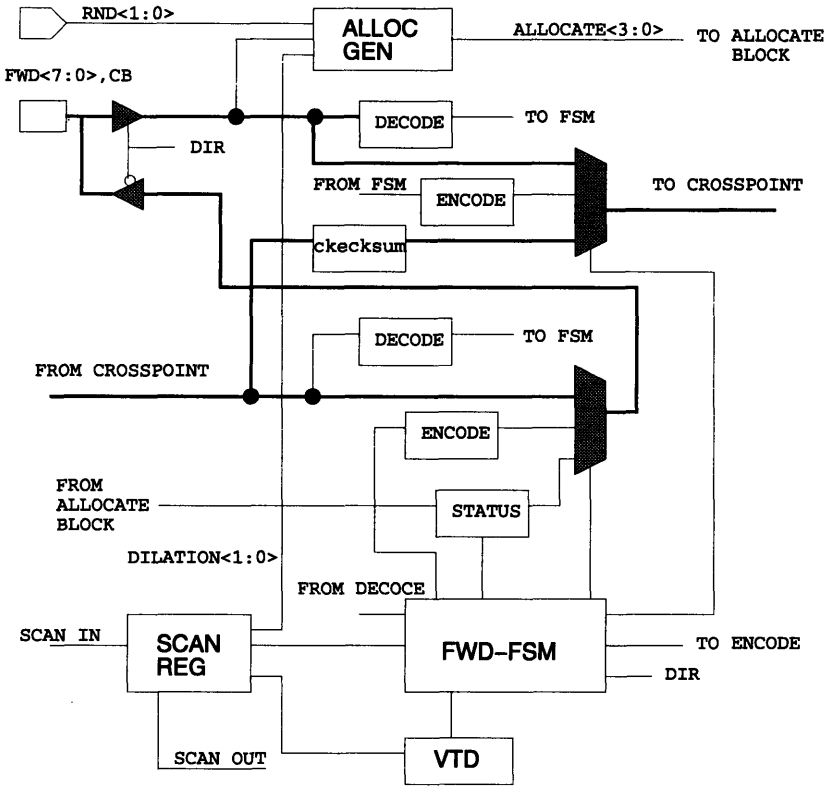
The diagram of the forward port is shown in Figure 4-2. The forward port is initially an input port and requests the allocate block to open a connection using the allocate request generator (ALLOC GEN) when ROUTE comes in. When the control word TURN comes along, the forward port becomes an output port after variable turn delay (VTD).

When TURN is received, the status and checksum are added. The checksum block is basically a linear feedback shift registers and computes $x^{16} + x^{12} + x^5 + 1$ as a cyclic redundancy code in CCITT. In the forward turn, the checksum block in the backward port is used in order to improve circuit observability.

Distributed Finite State Machines

The finite state machine (FSM) of the connection state transition in Figure 3-4 in Chapter 3 has some difficulty in being implemented, since a forward port can be connected to any backward port. To cut down a large number of signals which have to run across the chip, we implemented the FSMs in parallel decomposition as shown in Figure 4-3. This decomposition also leads to the reduction of the input and output signals of the FSM and contributes to circuit minimization.

[2]The value of 4 to 6 is reported.



Variable Turn Delay (VTD) tells when to expect meaningful data using the configuration register in the scan register.

Figure 4-2: Forward Port

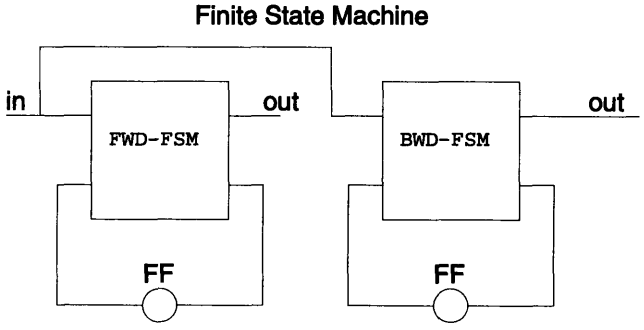
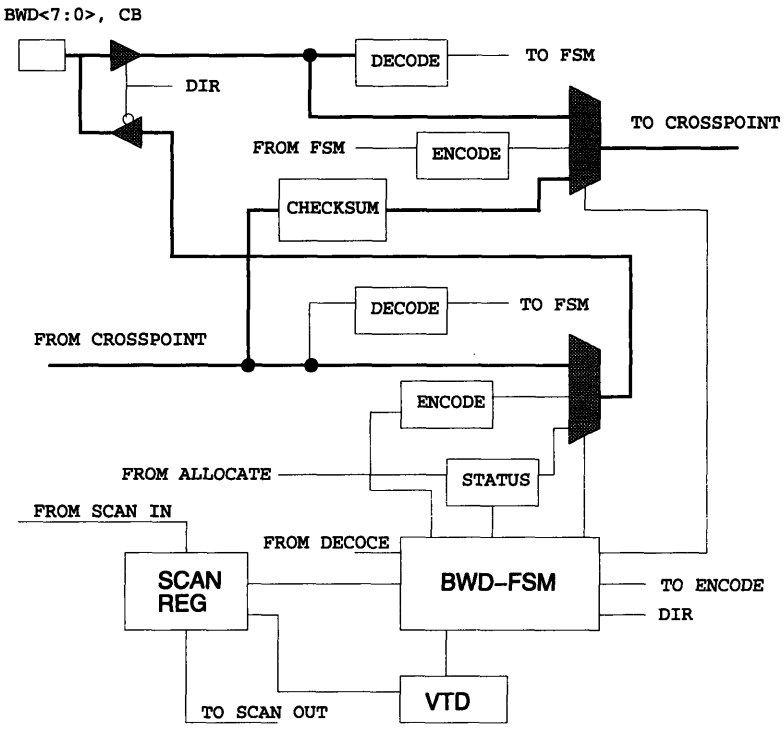


Figure 4-3: Parallel Decomposition of Finite State Machine



Checksum generates checksum data for health check and also serves as a chip testing provision of loop back.

Figure 4-4: Backward Port

4.2.4 Backward Port

The diagram of the backward port is shown in Figure 4-4. The basic structure is the same as the forward port except that there is no allocate request generator.

The backward FSM is simpler than the forward FSM, since there is no connection opening capability.

4.2.5 Crosspoint Array

The basic function of crosspoint is to select one of the input ports and transmit the data to the specific output port. Naive implementation is to put 4 n-channel transistors whose sources are connected together. Originally, to improve performance, an extra n-channel transistor was put in series to limit the voltage swing of the heavily loaded intermediate node. In this implementation, getting full swing output voltage was difficult, since the

output inverter operates in a high gain region and the trip point has to be shifted to ensure correct operations in all process corners.

Therefore, a differential scheme is chosen to exploit the fact that differential receivers are potentially faster than single-polarity receivers. Since the datapath has to run across the chip, minimal propagation delay is to be ensured by repeaters. As a result of fine tuning of the repeaters, it is possible to have both polarities of the signals in approximately equal delay. Using a differential amplifier and a matching trip point inverter makes the output signal full swing. Figure 4-5 depicts a differential multiplexor of the crosspoint array[3].

However, from the results of SPICE (level 28) simulations, an alternative crosspoint design without an n-channel transistor in series (1) was faster in this particular case. The reason was that the intermediate node is not loaded heavily enough in the case of a 4-input crosspoint and also that the bias voltage has to be carefully chosen. As a circuit scales, the design (2) will outperform the design (1).

The biasing scheme of the differential amplifier is similar to CMOS ECL receivers introduced by Chappell *et al.* [9]. The double-feedback biasing of both the load p-device and the current-source n-device provides good compensation for power supply and P-relative-to-N device common-mode shifts.

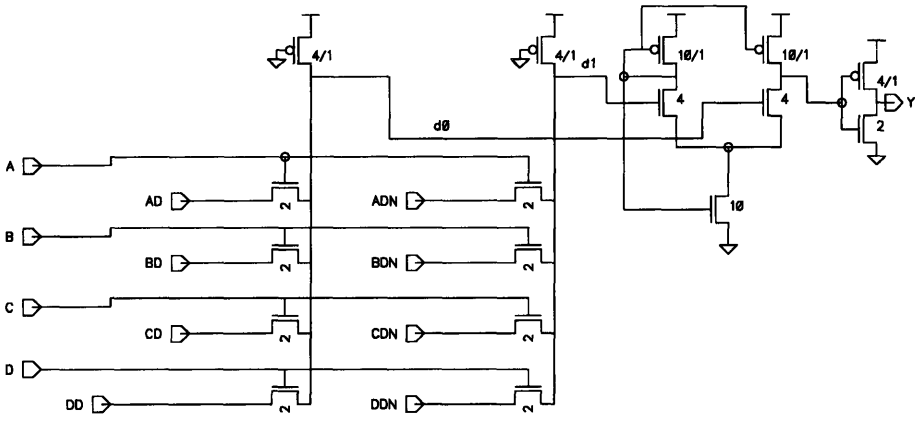
4.2.6 Allocate Block

The allocate block is depicted in Figure 4-6. Figure 4-7 depicts an allocate logic (ALC) of the allocate block[4]. The allocate block allocates the crosspoint array connections to requesting forward ports, which implies arbitrating the requests from the forward ports and turn on appropriate crosspoint switches.

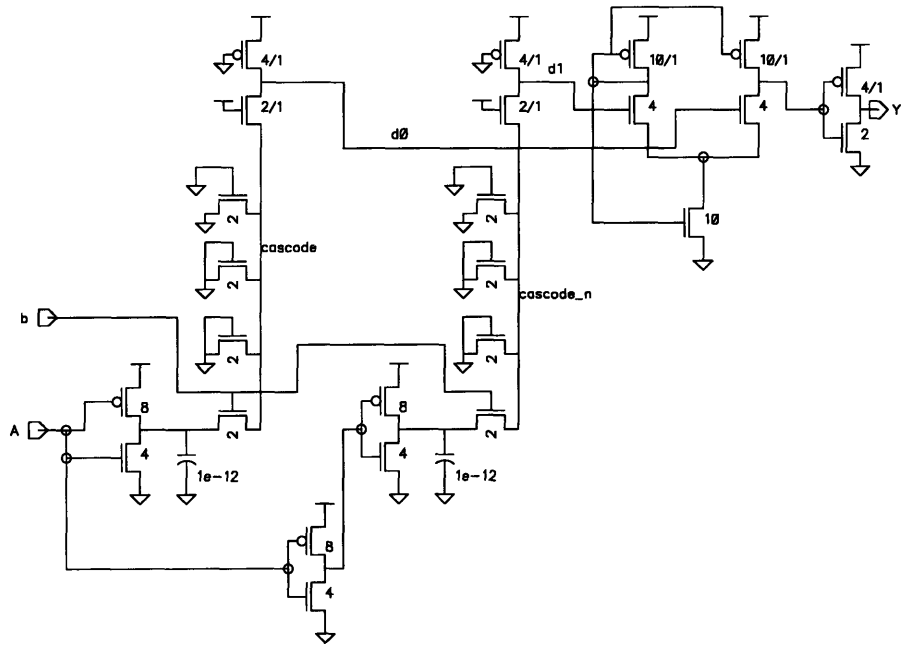
Low-power implementation of this allocate logic is depicted in a complementary manner in Figure 4-7 (1). Three output inverters are sized to ensure proper voltage level of output signals X, Y, Z, because p-channel transistors in series cannot pass voltage low without degradation.

High-speed implementation of this allocate logic is depicted in Figure 4-7 (2). This circuit consists of a receiver stage, a priority stage and a driver stage. The pull-up p-channel transistors in the receiver stage and pull-down n-channel transistors in the priority

[3]The layout of the circuit is found in Appendix A.
[4]The layouts of these circuits are found in Appendix A.



(1) Signals A through D come from the allocate block to open and close the connections. Signals AD, ADN through DD, DDN are datapaths of both polarities. As the number of inputs increases, the cascoded version performs better.



(2) The simulation circuit for a cascoded version. Bias voltage can be applied to the gate of cascode NFET to limit the voltage swing of the heavily loaded crosspoint nodes such as CASCODE and CASCODE_N in the figure, since with the gate of the series transistor being held at a constant voltage, the drain of the driver transistor will be held to a threshold below it. The series transistor, in effect, acts as a common gate amplifier.

Figure 4-5: Crosspoint Differential Multiplexor

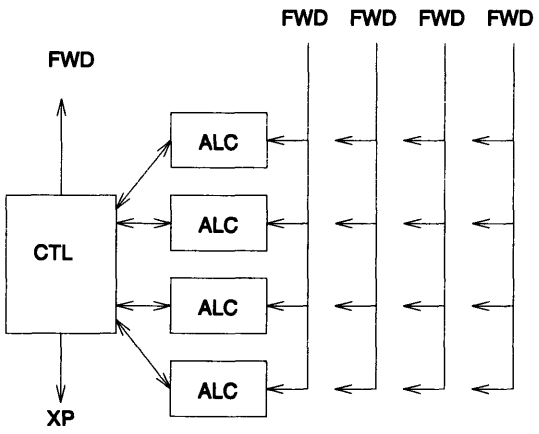


Figure 4-6: Allocate Block

stage are ratioed such that V_{IL} of last stage inverters is reasonably low. This ratio-circuit version is much faster and smaller in layout area than the complementary implementation, but incurs static power dissipation.

In the Hewlett Packard's 0.8μ effective gate-length process CMOS26, the simulations show the high-speed version achieves 0.6 ns delay with moderate loading.

However, as the number of input increases, the circuit size grows due to the pull-down transistors. With large number of inputs, Manchester carry chain style arbitration becomes more attractive [26].

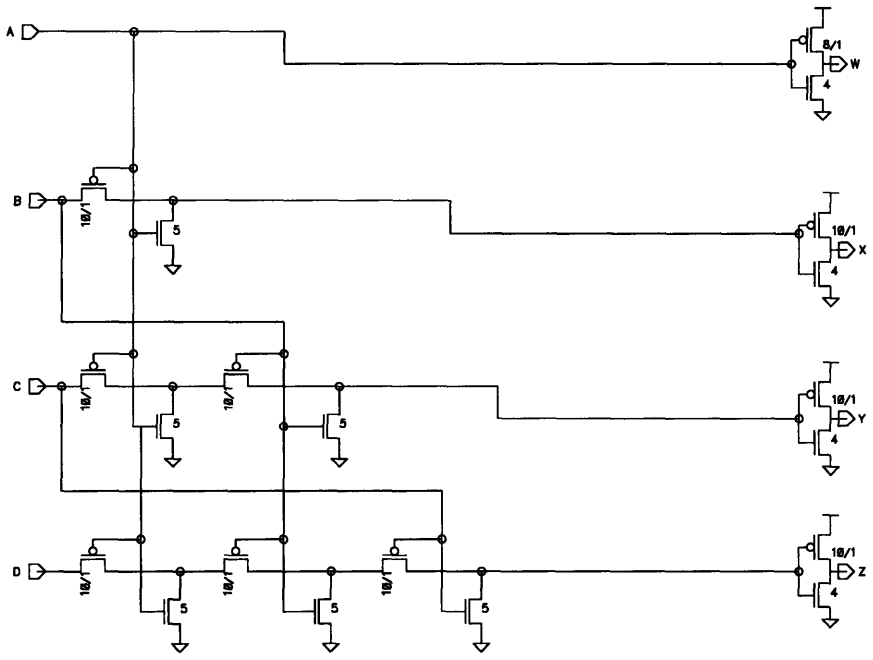
4.2.7 Test Access Port and Configuration Registers

The configuration of the chip is provided by way of TAPs depicted in Figure 4-8. The dilation mode, variable turn delay value, port enabling, fast reclamation mode, swallow mode, impedance control value[5] and signal delays can be configured.

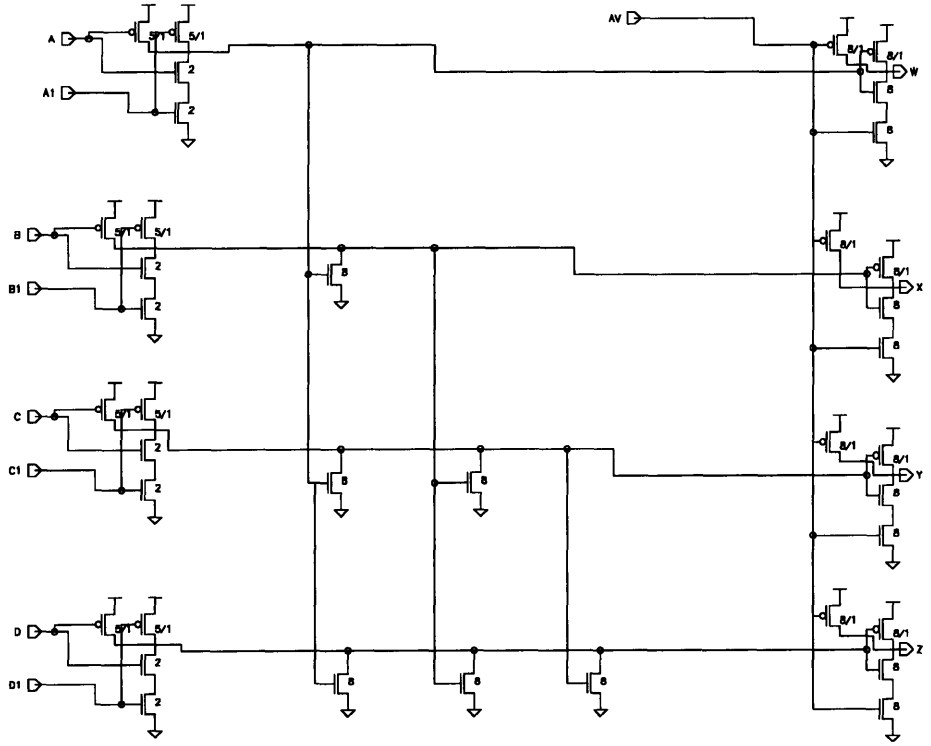
4.2.8 Pad Ring

The power dissipated for off-chip driving is very dominant in the total chip power dissipation and becomes even more dominant with scaling of a chip size. To reduce power dissipation and achieve low communication latency, point-to-point signalling with reduced voltage swing and matched impedance is essential. Since there is no DC power dissipation, series termination has an advantage over parallel termination.

[5]Section 4.2.8 describes this in detail.

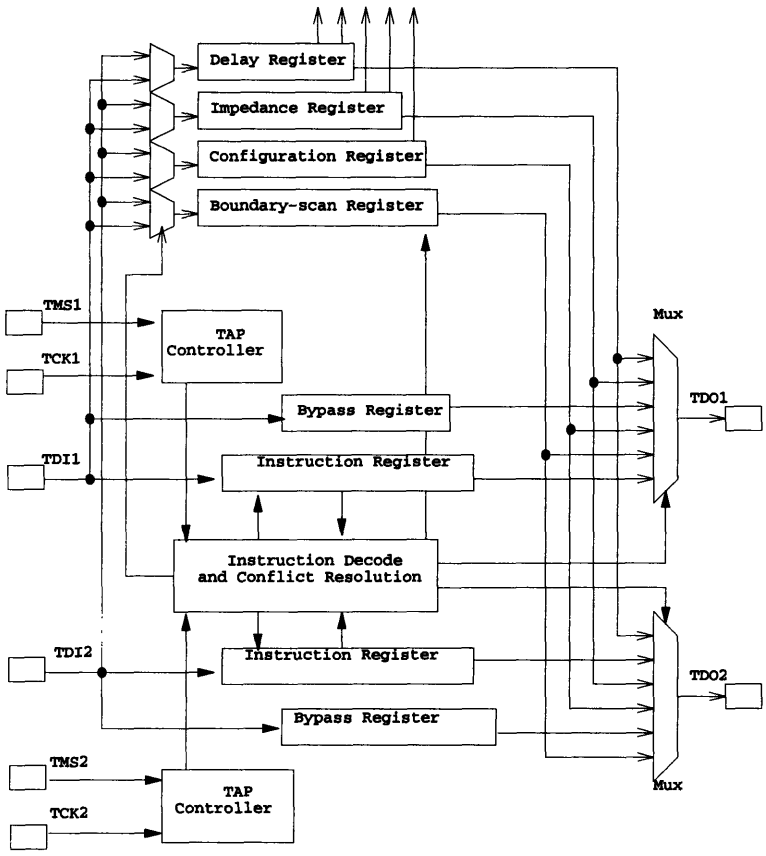


(1) Signals A through D come from the forward ports requesting the same backward port. The fixed priority is applied to these requests and the winner gets a grant signal.



(2) Signals A through D come from the forward ports that are requesting the same backward port. In this example, extra gating logic is included. The pull-down transistors have to be properly sized for correct operations.

Figure 4-7: Allocate logic



Each scan path has its own instruction register, bypass register, and TAP controller. The difference between single TAP and the multiple TAPs is an addition of conflict resolution unit.

Figure 4-8: Scan Architecture for Dual-TAPs

As long as the gate voltage can go one device-threshold higher than the signalling high voltage, n-channel devices can be used as pull-ups of the drivers. The n-channel devices have advantages over p-channel devices in speed and size; therefore, the driver constructed with only n-channel devices achieves lower output capacitance, lower internal capacitance, smaller layout, faster operation, lower power dissipation, and good latchup prevention[6].

As for impedance matching, [16] introduced an on-chip impedance control pad for reduced voltage swing transmission. Figure 4-9 shows an example of such a bidirectional pad. The impedance control register in TAP controls the pull-up and pull-down networks to match the characteristic impedance outside the chip. The impedance selection algorithm is described in [14]. The receiver circuitry is a cascode amplifier. When the input signal falls one-threshold below the reference voltage, the input transistor conducts, pulling down the input to the inverter below its trip point. When the input signal rises, the input transistor shuts off, and the current source (grounded p-channel transistor) pulls the input of the inverter up to the supply voltage.

The total I/O delay of this bidirectional pad in CMOS26 process accounts for only one-third[7] of conventional 5V full-swing pads.

To adjust the delay for making the signal transmission time a multiple of a clock cycle, an adjustable delay circuit is employed as depicted in Figure 4-10. The signal delay is controlled by serially connected inverters and RC load similar to the voltage-controlled delay line [17]. Note that in this circuit the capacitance is split into two moving it toward both rails to sharpen the edge rate equally for the rising and falling.

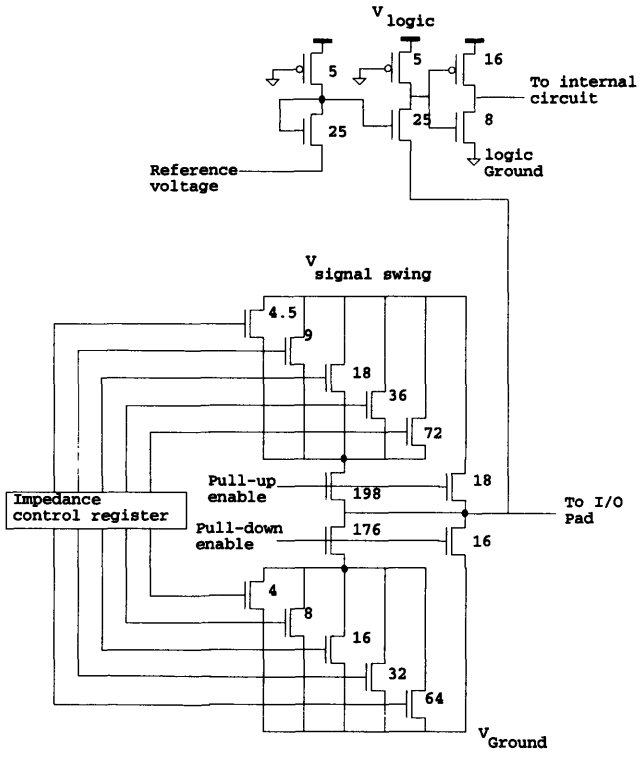
Alternatively, if we were short of I/O pins, we could go into full-duplex bidirectional pads. Dally, Dennison and Lam introduced a current steering bidirectional pad [19]. The full-duplex bidirectional communication will extremely simplify the protocol and reduce the size of FSMs.

4.2.9 Global Routing

The global routing is preferably done in thick metal layers rather than thinner layers such as metal-1 and metal-2, whereas the local routing is done in poly and metal-1. When necessary,

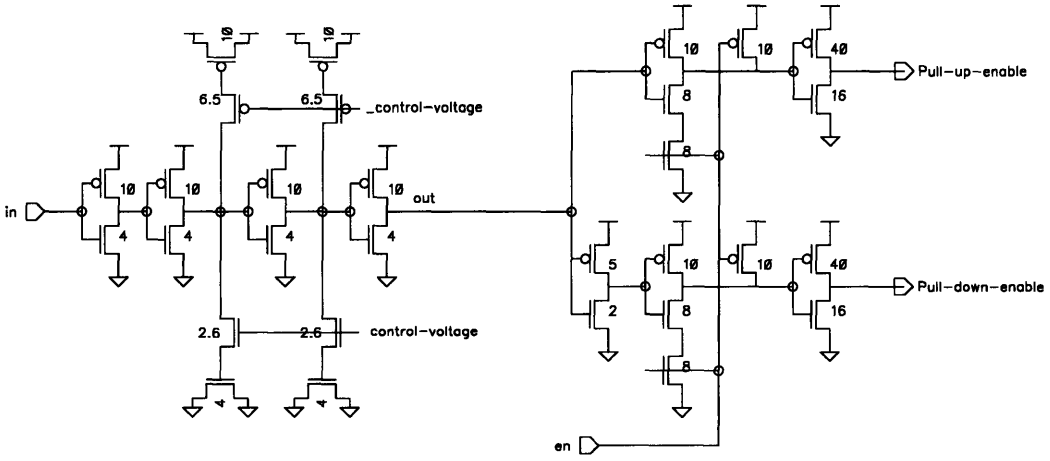
[6] If a p-channel transistor is used and the input voltage becomes higher than $V_{signal-swing} + |V_{be}|$, the leak current flows from the input terminal to the back-gate of the p-channel transistor.

[7] The driver approximately consumes 2 ns and the receiver consumes 1 ns, whereas the conventional pads consume around 10 ns.



A bidirectional pad contains both low-voltage automatic-impedance-controlled driver and receiver circuitry. Shown here is a typical bidirectional pad configuration. The protection circuit is not shown for the sake of clarity. Note that pull-up devices are slightly larger to compensate for the difference of $(V_{GS} - V_T)$, since the operating voltages of the sources differ and the body effect changes V_T at the same time.

Figure 4-9: Bidirectional Pad



An adjustable delay circuit feeds a signal into a pad driver circuit, which scales up to drive a large load. The control voltage of the adjustable delay circuit typically comes from a D/A converter which generates a bias voltage from the values in the delay register in TAP.

Figure 4-10: Pad Driver with Adjustable Delay

wide wires are used to reduce the interconnect delay, since the resistance decreases linearly with the wire width while only the area capacitance portion increases and the relative fringing field effect diminishes[8].

Clocking

A single-phase clocking scheme with edge-triggered flip-flops is used in the chip design. The main reason is because clock distribution becomes considerably easier than two-phase clocking. The amount of signals that have to be distributed all over the chip is reduced by adapting a single phase clock scheme. Problems with clock pulse-width narrowing, and race conditions associated with a single-phase clocking with latches[9]can also be avoided with a little performance and area penalty. The other reason is because guaranteeing the nonoverlap time for a two-phase clocking scheme over all process corners can cut significantly into the cycle time.

From our experiences, the hardware description language (HDL) codes were good in

[8]However, large buffers are needed to drive large capacitance. Overall delay time has to be evaluated by simulations.

[9]The problem is caused by data transparency during the clock cycle; therefore, the clock skew and the gate delay have to be managed in every logic stage. This problem can be alleviated by distributing the clock in the opposite direction to the data stream, but this does not guarantee correct operation all the time.

terms of efficient implementation of specification, but the clock input of each module had to be one pin[10]. Therefore, the clock signals had to be buffered inside each module, which incurs clock skews, since the load of each clock signal is different from one module to another. The partial solutions to this problem are either to feed a clock signal which is faster than others to a certain module that has heavier clock loading or to edit synthesized schematics and create equally loaded clock input pins.

The clock buffer consists of cascaded inverters scaling up approximately three times in each stage except the last stage. The final inverter size is calculated to satisfy the transmission line termination condition. The inverter before the final one is slightly larger than the final inverter to achieve a sharp waveform.

Clock distribution is done from the center of the chip outward following the ground and power distribution tree. Ideally, the H-clock tree with line width reduced at each branching point for proper impedance matching is desirable to reduce the clock skew[11] [20, 21]. Alternatively, in the datapath where intentional skew is sometimes needed, a powering clock tree is suited. The stage-connecting-wire scheme[12] is reported to reduce the clock skew down to one-third in a 500 MHz clock distribution tree [29].

Power and Ground Routing

The global power/ground routing uses wide metal-3 as much as possible because metal-3 is thicker and farther from ground than the other metal layers so both resistance and capacitance are lower per unit length; therefore, it can avoid the voltage drop and sustain noise margins[13].

In other layers of ground and power supply distribution, jogs of wires should be avoided as much as possible to avert any electromigration problem[14] and to reduce the wire inductance. Depending on the length and number of rows (accumulated I_{ds}, to be precise), the wire width should be increased to keep maximum current density less than the rule of thumb value, 1mA/square micron, and to reduce the voltage drop.

[10]The real reason is the discrepancy between a functionality block unit and a physical clock distribution unit. Usually, this is not a significant problem in cell-based designs.

[11]The process variation is less severe for wires than transistors.

[12]The outputs of the buffers in the same stage are connected by wire to equalize the clock skew.

[13]Also, if necessary, on-chip bypass capacitors can be placed under metal-3 power lines to reduce power-supply noise.

[14]This should be considered at the worst-power corner including the effect of power supply noise.

The power/ground supply nets to internal circuitry and off-chip drivers have to be separated to avoid disturbance of the internal circuitry (fluctuations at the power lines) that will otherwise be caused by the simultaneous switching of output buffers.

In high-speed designs, power and ground cannot be treated as DC signals but rather AC signals, since transistor switching noise creates a considerable amount of noise in power supply lines. They should be considered to be return paths of transient current when switching occurs.

4.3 Hot Spot Avoidance

4.3.1 Problem Statement

When the dilation is configured to be one, the routing method of the Transit network becomes fully deterministic in that the path a message follows is determined only by a source node and a destination node.

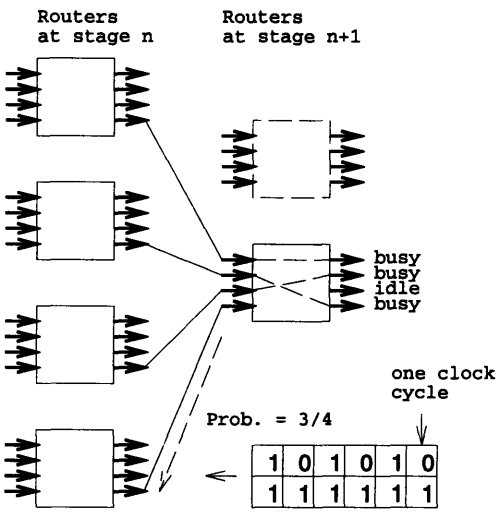
When the dilation is configured to be more than one, the routing method of the Transit network becomes *oblivious* routing in that a router can choose a path among different paths, but without global information about the network state.

In either case, the source-responsible protocol described in the previous chapter suffers from performance degradation if there is a hot spot in the network. In general, Pfister and Norton showed that the presence of hot spots can severely degrade performance for not only traffic due to synchronizing processors, but also all traffic in MINs [22].

4.3.2 Solution

The scheme which is employed here is similar to adaptive routing. The probability of getting busy from a downstream router to an upstream router is to be propagated back (See Figure 4-11) so that the upstream router can have some information of a hot spot in the network. Then, this information effectively prioritizes paths among maximal-fanouts and allows the upstream router to choose an alternative path to alleviate the congestion.

The probability of an incoming message's getting blocked can be assessed by the ratio of free output ports to used output ports. Each output port has an IN-USE bit that represents whether the output port is being used or cannot be used for an incoming message because of a faulty downstream router. All the free input ports propagate back the probability to



A router at stage $n+1$ propagates back two IN_USE bits in every clock cycle repeatedly by way of free links. The sequence does not matter, but logically equivalent ports are transmitted in the same cycle.

Figure 4-11: Hot Spot Avoidance

upstream routers by way of backward control bits (BCB0 and BCB1) that are used for fast path reclamation and backward port disabling (See Table 4.2). Since only free links are utilized, there is no need for extra hardware cost.

The output ports of the upstream routers accumulate the serially transmitted probability and use this information to decide which path to choose among logically equivalent paths. (See Figure 4-12) Since our protocol is source-responsible and the penalty of being blocked in later stages is rather large, avoidance of being blocked is especially beneficial to overall performance.

Note that the the serially transmitted probability does not slow down crosspoint allocation. Because we do not have to distinguish each bit, a downstream router can simply send it recursively, and an upstream router accumulate it in a fixed time window and use this information as approximate priority at the time of allocation.

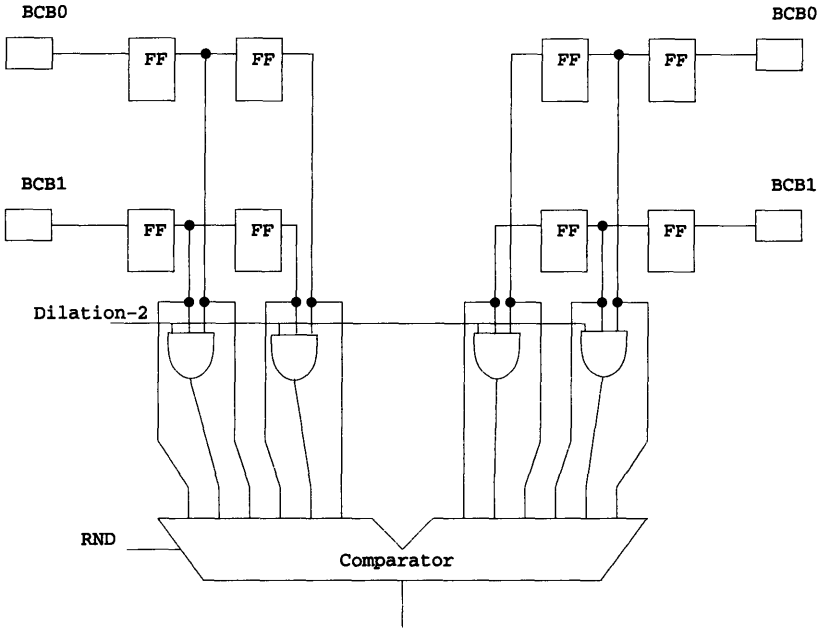
4.3.3 On-Line Dynamic Reconfiguration

As a network size becomes larger, mean time to failure (MTTF) of the system gets smaller. To build a practical system, faults in the system should be tolerated and contained.

In general networks, Leighton and Maggs introduced a theory for on-line reconfiguration

Signals	Usage	Polarity
BCB0	fast path reclamation while in use hot spot avoidance while idle	active low high: busy
BCB1	backward port disable hot spot avoidance while idle	active low high: busy
IN-USE	backward port in use	high: in use

Table 4.2: Signals for Fault Propagation



IN_USE bits are fed into shift registers for priority comparison. In the case of dilation-2 mode, a blocking state of logical paths is incorporated as an extra bit in the decision making.

Figure 4-12: Hot Spot Avoidance Circuit

[25, 24]. Chong explored analog-controlled adaptive routing in [30] and also introduced the simplified algorithm of Leighton and Maggs [10]. However, those schemes need a central entity which can control all routers or a distinct network from the one in which messages are delivered. Hence, the cost of hardware is substantially large. Also, analog circuits suffer from variation of power and ground in the system, and that may lead to malfunction due to reduced signal to noise ratio[15].

On-line dynamic reconfiguration, on the other hand, only needs a local decision and no extra network is necessary. This scheme intends to provide a cost-effective method to alleviate the problem locally as a quick-fix or first aid before network-level reconfiguration is required.

Before we discuss faults in a network, fault model should be defined. We assume that faults occur either in routers or in wires without correlation and that there is always at least one TAP controller operational in each router.

When width cascading is employed, a fault detection scheme is needed to prevent messages from splitting because of faulty routers or flipped bits. Figure 4-13 illustrates this message splitting. A discrepancy of IN-USE bits results if one router misroutes a message. Then, the routers report to the upstream routers to drop this message by sending BCB0 and BCB1. The faulty routers are marked faulty in the corresponding upstream router's output port controller and are eliminated from the choices it can make.

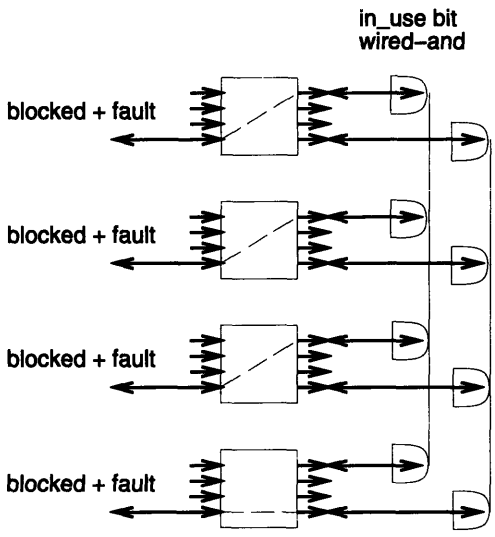
After isolating suspects for faulty routers or links, a health check message has to be delivered fully deterministically through the specific routers to check the sanity of these deactivated routers and links. Because there might be a possibility of intermittent errors of links, those routers and links cannot be determined faulty until this is confirmed by a health check message.

One way of conducting a sanity check is to configure the entire network to dilation-1 by way of TAP, and launch a health check message with a fully specified destination address[16] in order to deterministically route it through intended routers.

This could also be done within a set of routers without stopping the entire network, while other routers are operating in another dilation mode. First, all ports of the selected routers

[15]When analog circuits are integrated in digital circuits, they have to be protected by shielding or separate power lines from noise caused by switching of digital circuits.

[16]The number of bits consumed in each router changes depending on the dilation mode. To fully specify the route from a source to a destination, extra bits are needed besides a destination address.



When misrouting is detected by wired-AND of each IN_USE bit, BCB0 signals to the upstream routers to drop the connection mimicking blocked routing. At the same time, BCB1 signals to disable the output port until a health check is conducted.

Figure 4-13: Fault Detection

are disabled and the routers are configured to dilation-1 by TAP. Second, the ports which are going to be used are enabled. As long as health check messages and ordinary messages are not mixed up in the routers under sanity checks, this procedure can be conducted in parallel to normal operations.

Such on-line configuration and health check techniques will become more critical as the network size scales up, since the cost of reconfiguration (down time) goes up accordingly.

4.3.4 Procedure Implementation

The implemented procedure is as follows:

1. While the link between the input port of a router and the output port of the connected router is idle, IN-USE bits of all output ports of the downstream router are serially transmitted to the upstream router repeatedly in BCB0 and BCB1. To alleviate the effects of stuck-at-faults, the polarity of each signal has to be chosen carefully. Table 4.2 explains each signal in detail.

2. The output ports of the upstream routers accumulate the serially transmitted probability. In choosing among logically equivalent paths, the upstream router compares

each probability and chooses the most unused downstream router. If the probability is the same, random bits are used to decide.

3. When the state of the link changes to FORWARD, the output port disregards the probability. After variable turn delay, it treats BCB0 and BCB1 as fast path reclamation and backward port disable. When misrouting is detected by wired-and of each IN_USE bit, BCB0 signals to the upstream routers to drop the faulty connection by mimicking blocked routing. At the same time, BCB1 signals to disable the output port as a suspect until a health check is conducted.

Even though the propagated probability does not accurately correspond to a congestion in the network and steering of a message can be done only one-stage upstream, this procedure is a cheap way to resolve the problem of oblivious routing. Moreover, this scheme does not incur any increase of communication latency nor any increase of protocol complexity.

In conjunction with fault detection, the messages are effectively managed to stay away from a troubled spot, whether it is caused by temporary congestion or a faulty part.

4.3.5 Circuit Implementation

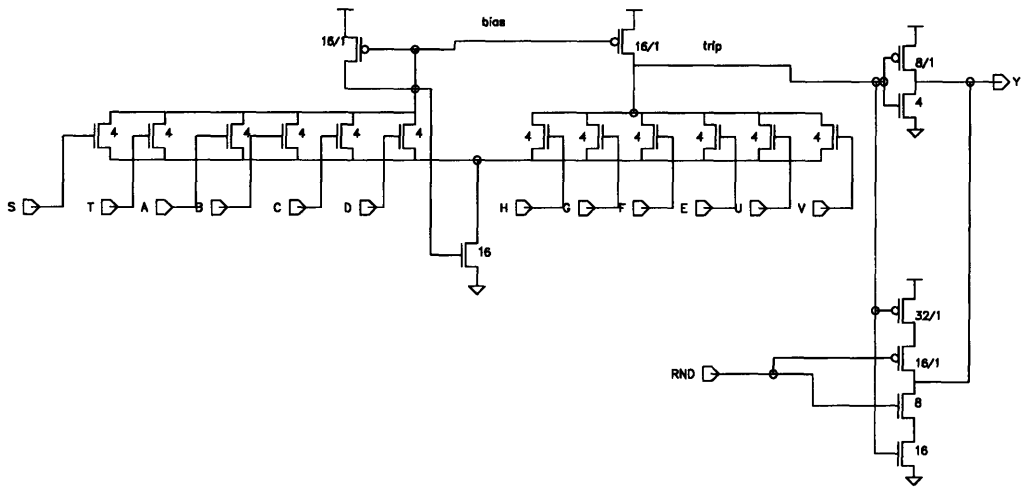
Figure 4-14 shows a circuit implementation of priority comparison function. Note that a randomly generated bit is used to determine an output only when the two priority inputs are even. This, in effect, serves as stochastic path selection. Some of the results of SPICE simulations are found in Appendix A.

4.4 Areas for Improvement

1. Retransmission Policy

   When a message is blocked in a router, checksum and status information are sent back to the source node. The checksum data can show whether the path that the message traversed was erroneous or not. The status information can give some information as to how congested the network was.

   If any source node is not allowed to send messages to other nodes other than the destination as an intermediate node in a multihop scheme, the source node has two choices in retransmitting a message. One is to wait for a while and retransmit the



Two priority inputs are compared using a double-feedback-biasing differential amplifier whose output level matches the trip point of the output inverter. When in a draw, the random bit is used to decide the output by a parallel driver properly sized for correct operations.

Figure 4-14: Priority Comparator among Maximal Fanout

message. The other is to retransmit it as soon as possible. This policy is highly dependent on applications, but we might want some support in the network to estimate waiting time in order to improve overall performance, since in the multithreaded architecture, scheduled waiting time is not a penalty at all.

2. Synchronization Support

   Synchronization variable access can severely degrade network performance. Therefore, software combining or some measures to reduce hot spots is indispensable in MINs. One hardware method is wired barrier synchronization scheme, which has a separate network used only for barrier synchronization to achieve low overhead. The drawbacks of this scheme are that the synchronization pattern has to be predicted at compile time and that the bus-based wired-NOR barrier line inherently does not scale with the number of processors.

3. Performance Monitoring

   From the standpoint of software development, it would be beneficial to have performance monitoring and debug support engineered directly into the hardware.

4.5 Summary

In this chapter, METROSC implementation is described. With low communication latency and fault-tolerant provisions, the chip will achieve substantial performance gain. However, building a large crossbar switch is still a challenge limited by electrical constraints. The partial solution[17] to this problem at this moment is a hierarchical network construction presented in Chapter 3.

[17] Optical interconnect seems to be a promising technology to build a large crossbar switch.

Chapter 5

Design Verification and Testing

The chip design was done in register-transfer-level (RTL) descriptions in an HDL called Transit Canonical Form (TCF) [32] to make the design specifications independent from any HDL tools. This TCF descriptions were transformed into Verilog HDL descriptions and used in the simulation and synthesis. In this chapter, design validation of the netlists and testing of the chip are discussed.

5.1 Design Validation

As chip size scales up, the challenge of capturing large amounts of functionality and getting it correct becomes overwhelming. It is not possible to know if the behavioral code is completely validated. One way to get around this problem is to construct a testbench similar to the real system as accurately as possible and to test all possible test combinations. The closer the testbench looks to the real system, the better chance we have to achieve correct designs.

A well-constructed test provides informative log messages if sophisticated self-checking codes are incorporated. It is important to automate some checking process, anticipating many iterations of running the same test vectors.

5.1.1 Simulation Model

Different simulation models are used depending on the level at which each simulation tries to verify. We move up the simulation hierarchy as modules[1] become verified at lower

[1] A module here refers to an entity which has distinct functionality from others and whose layout is usually done in the proximity.

levels. For instance, at the lowest submodule level, SPICE simulation is done for timing and functional verification. Then, at the module level, interface signals are fed into the module by a waveform editor and verified by observing waveforms of output signals. For the modules that contain FSMs, it is also necessary to monitor state encoding signals. Finally, at the chip level, all the modules are connected and the input signals are fed by a pseudo-system model such as illustrated in Figure 5-1. This model is written in Verilog HDL and C. For the purpose of functional simulations, the special modules that are designed at the transistor level have to be described in RTL. These models are verified prior to the chip simulation.

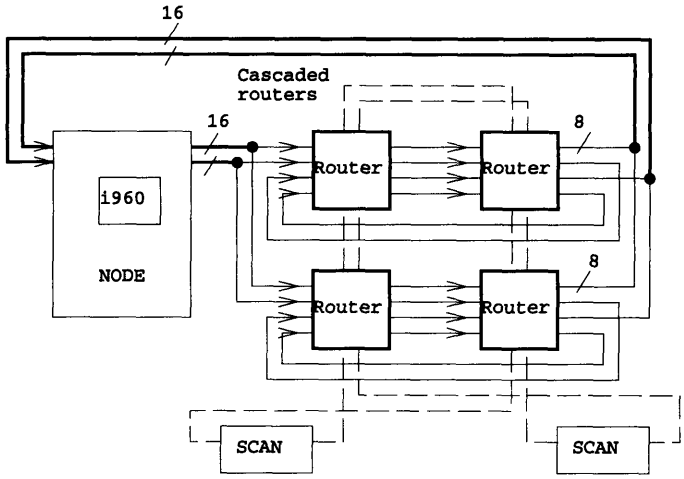
System Simulation Model

The system simulation model mimics a network by connecting the backward ports to the forward ports of the subsequent router to verify the functionality. Figure 5-1 illustrates the system simulation model. The processing node contains the Intel 80960 hardware verification processor model which executes commands in the Logic Automation Processor Control Language (PCL) and represents the processor within the simulated design by simulating bus cycles. The processing node model is depicted in Figure 5-2. The interface of the node processor with the network is handled by Net-In [34] and Net-Out [33] which are connected through the pipelined internal 64-bit bus to the processor.

The system-simulation testbench is constructed in the Verilog environment that consists mainly of three parts: the initialization and configuration, the monitoring and error checking, and the simulation control. The scenarios which the simulation follows are written in PCL, and the codes generate message traffic through the constructed network. By using PCL, the processor can configure Net-In and Net-Out, inject or abort network operations, and check the status of the ongoing or recently completed network operations. Depending on the contents of the simulation, different system simulation models and configuration files are used to verify the design.

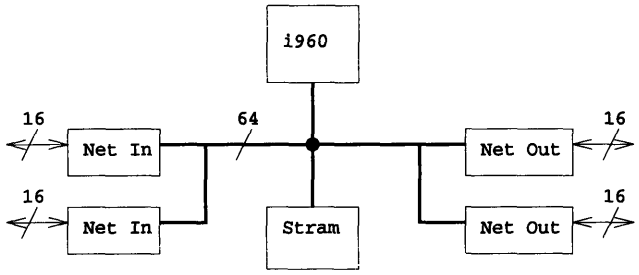
5.1.2 Test Vectors

In the system simulation, test vectors to the router are generated by PCL codes in the pseudo-system simulation model. These test vectors should be as rigorous as possible. Arbitrary combinations of test vectors are necessary to shake off bugs in the design, since



In this example, two routers are cascaded resulting in a 16-bit datapath. A dual scan path scheme is employed. A four-stage network is mimicked by this model. Note that the directions of datapaths are for initial data injections and can be independently reversed by TURN.

Figure 5-1: System Simulation Model



STRAM stands for a synchronously timed SRAM. Net-In and Net-Out are network interface chips designed by André DeHon.

Figure 5-2: Processing Element Model

it is often the case that bugs come in from designers' blind spots.

Simulation invocation is automated so that only newly changed modules are compiled and simulation is invoked properly. This also helps improve the verification productivity. Whenever a change to a module is made, a set of regression tests are run to verify correctness.

The checking tools are incorporated in the simulator in PCL codes so that tedious error checking of the simulation results can be avoided[2]. The output of the checking and simulation progress reports are stored in a log file in a format which computer programs can easily parse for automated verification. Only crucial parts of the results are condensed to a note file that will give minimal information about errors and pointers to a log file.

5.2 Testing

5.2.1 Manufacturing Test

Manufacturing test vectors can be derived from the simulation results, but they have to be condensed because of the limitation of tester requirements. Ideally, we like to activate the circuit 100 percent and to propagate the results out to see if all circuitry works. To generate effective test vectors, the controllability and observability of the circuit have to be improved.

5.2.2 Circuitry Facilitating Chip-Testing

Since the advent of ASIC, the desire for reliable VLSIs and rapid time-to-market has forced the test problem earlier in the design cycle. It is becoming increasingly important to think how to test a chip and to design in testability early in the design cycle. For instance, to make simulation effective, we have to pay attention to the observability of the circuit. Otherwise, the test vectors may easily exceed the practical limit of simulation time, or, even worse, it would become impossible to test the whole circuit.

The controllability and observability of the circuit are improved by scan-based TAP interface. For instance, to minimize the test vectors of sequential circuits, the initial states of large FSMs can be set by the scan interface. After some testing of combinational circuits, the state encoding can be read out from TAP to strengthen the observability of the circuits.

[2]Since launched messages are received by the same node, the checking is trivial and easily automated.

When a full serial-scan is impractical, even a partial scan can be very helpful in testing. Extra circuits required for scanning can be placed selectively so as not to affect critical paths.

5.3 Future Improvements and Observations

The allocate block is not scalable for a large number of inputs to a crosspoint array. One way to cope with an increase of inputs is through partitioning. The allocate operation can be sped up by partitioning and using a tournament-style structure. In the crosspoint array, the delay of row direction is overcome by proper repeaters, and the column is divided in the same manner as the allocate block. Then, the output of the proper division of the column is selected by the result of the tournament.

If a process that enables the cells to be covered with routing is combined with a good automatic placement tool, very high density of layout can be achieved. With a library in which the size of the transistors is parameterized and some generator support for regular-array structures, such as datapath, is available, the density can rival that of a full-custom design.

As the cycle time continues to be reduced, we will face tester limitations, and on-chip test circuitry such as Built-In Self Test (BIST) will become necessary to match the speed of the chip. Also, for the testing at the wafer level, generating the test vectors on-chip avoids the difficult problem of transmitting synchronized very-high-speed signals through probe impedance discontinuities.

At the system level, BIST, after power-on reset, would also be helpful to determine component failures in a large system. By obtaining the results by way of TAP or other dedicated hardware, the system can change its configuration promptly before the system starts operation.

When a complicated ASIC is developed, to fully test a chip in itself is a challenge. Incorporating Automatic Test Pattern Generation (ATPG) is helpful in effectively reducing manufacturing-test-vector-production time. A rigorous test-vector-generation program that can test all combinations of test vectors automatically would be a powerful aide for simulations.

For the system simulations, including "demon" models that could be programmed to

pseudo-randomly generate events, such as the message launch of a processing node or the injection of errors, would help to find bugs in the designs.

We used self-checking codes at the chip boundary, but code assertions at the module boundaries can give more precise checking capability. The usage of code assertions, which is a technique that operating system vendors, compiler writers and those who designed large complex systems have found useful in the past, exploits the fact that the designers can capture the specific knowledge they have about some boundary conditions inside the module when they are writing RTL codes. In the simulations, these boundary conditions are checked automatically and an alerting message will be printed out when they are violated, such as multiple bus drivers, protocol violations, and invalid states. These codes are kept to a minimum and cover design aspects not easily checked by comparing state information.

Coverage tools can also be deployed in the validation to know which sections of the overall design are being very well tested and which ones need extra coverage. This allows us to steer our efforts toward the places that need improvements of test vectors, which is a much more efficient way to use resources.

Chapter 6

Conclusions

This thesis presented a practical implementation for scalable multiprocessor-interconnect component designs. Issues of our concerns which are involved in developing scalable processors are low communication latency and fault tolerance. Since our approach seeks for a low-overhead unreliable protocol, the chip under development intends to offer a minimal set of functionality tuned for low communication latency and fault tolerance.

We find a full containment of network errors in hardware can lead to increased communication latency. In the past, the ARPA net was supposed to guarantee reliable message delivery without TCP/IP by the network hardware, but it turned out to be expensive and error prone. The degree of reliable message-delivery support of the hardware has to be carefully evaluated in a given technology.

When technology is rapidly advancing, delaying designs to seek for complexity is not a good idea. Simple, modular design is best, if possible at all. Using efforts only where crucial designs are needed is likely to yield cost-effective designs. Ideally, full-custom designs can take full advantage of what the technology offers at any point in time. However, the design complexity it will impose on designers is increasingly demanding. As systems becomes more complicated and a chip size gets larger, the productivity of the chip designers has to go up accordingly. For instance, behavioral synthesis of datapaths and regular arrays would be promising to improve productivity.

Regular structures, such as datapaths, and crucial parts are suited for full-custom designs, whereas the random logic of control signals can be cost-effectively implemented in cell-based designs. Some CAD tools start to support optimization of gate sizing of cell-

based designs for high performance. Thus, hybrid designs exploiting available tools and a finely-tuned parameterized library to improve productivity and performance will be proven to be a key methodology of VLSI design.

Appendix A

Datasheet

A.1 Signalling

The chip-signal decomposition of METROSC is tabulated in Table A.1. The random output signal is used in router width cascading. In order for all cascaded routers to choose the same output port among equivalent paths, one of the random output signals is shared by the cascaded routers.

A.2 Encoding

The encodings of the control words of METROSC are tabulated in Table A.2. The encodings are such that the detection of ROUTE and TURN is fast, since these control words initiate the critical paths of METROSC.

A.3 Layout Examples

Some of the circuits presented in Chapter 4 in this thesis are shown in Figure A-1, Figure A-2 and Figure A-3. Figure A-4 depicts the minimum-sized edge-triggered static D-FF in the standard-cell library developed for METROSC.

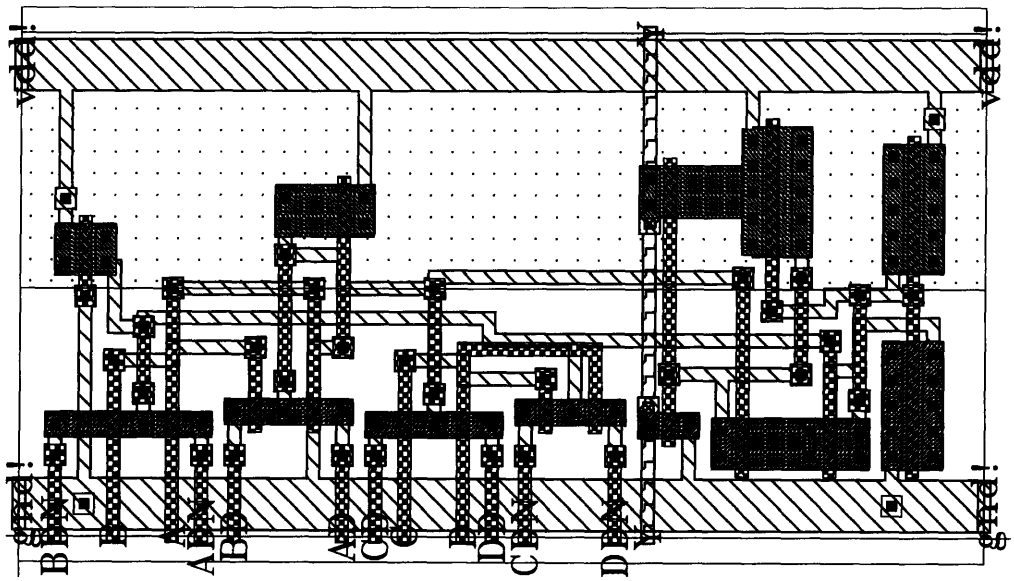
The standard-cell library was developed in CMOS26 process, which has three metal layers and one polysilicon layer. As a convention, input signals of the library cells are in poly layer and output signals are in metal-2. Power and ground are supplied by metal-1. Each row of standard cells has 12μ wide metal-1 V_{dd} at the top and Gnd at the bottom. We followed a

Signals	Number of Pins	Usage
Test Access Port TMS, TDI, TDO, TCK	4×2	system configuration and testing
Forward Port CB, BCB0, BCB1, D<7:0>	$(8+3) \times 4$	link connection
Backward Port IN-USE, CB, BCB0, BCB1, D<7:0>	$(8+4) \times 4$	link connection
Random Output	1	equivalence path selection
Random Input	1	equivalence path selection
CLK	1	clock input
Reset	1	reset input

Table A.1: Pin Summary

Logical Signal	Port Bits	
	control bit = CB	data = <7:0>
IDLE/DROP	0	-00
ROUTE	1	$-r_1 r_0$
TURN	0	-11
STATUS	1	in_use<3:0>
CHECK0	1	checksum<15:8>
CHECK1	1	checksum<7:0>
DATAIDLE	0	-10
DATA	1	—

Table A.2: Control Word Encodings



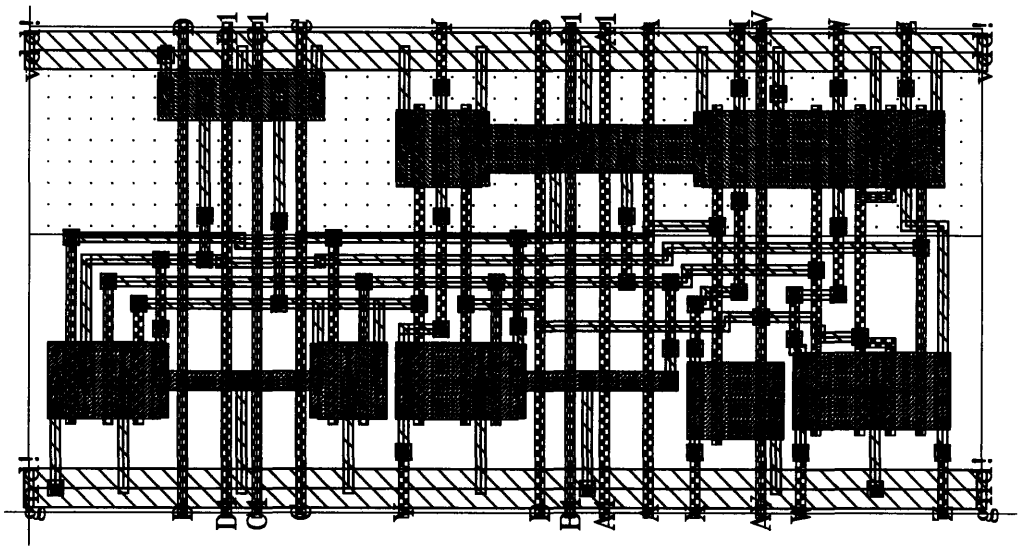
Signals A through D come from the allocate block to open and close the connections. Signals AD, ADN through DD, DDN are datapath of both polarities. As the number of inputs increases, the cascoded version performs better.

Figure A-1: Crosspoint Differential Receiver

convention that metal-1 for horizontal wiring and metal-2 for vertical wiring.

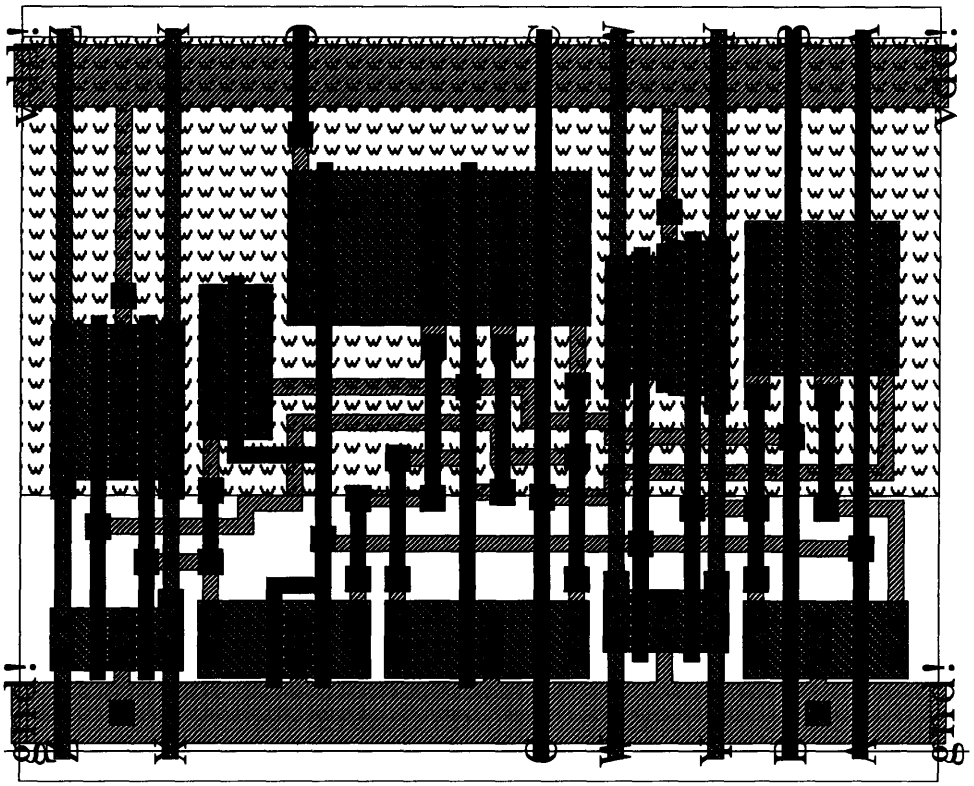
A.4 HSPICE Simulation Results

Some of the simulation results of the priority comparator depicted in Figure 4-14 in Chapter 4 is shown in Figure A-5. Only a random bit input RND, an intermediate signal TRIP, and an output signal Y are shown for clarity. Note that only in the case of tie where the input priorities are equal does the signal TRIP result in staying in half V_{dd}, and a random bit affects the output.



Signals A through D come from the forward ports that are requesting the same backward port. The other signals are used for gating logics incorporated for fast operation. The fixed priority is applied to these requests and the winner gets a grant signal. The pull-down transistors are properly sized for correct operations.

Figure A-2: Allocate logic (high speed)



The gating logics are not incorporated in this version, which yields small layout area and low power dissipation.

Figure A-3: Allocate logic (low power)

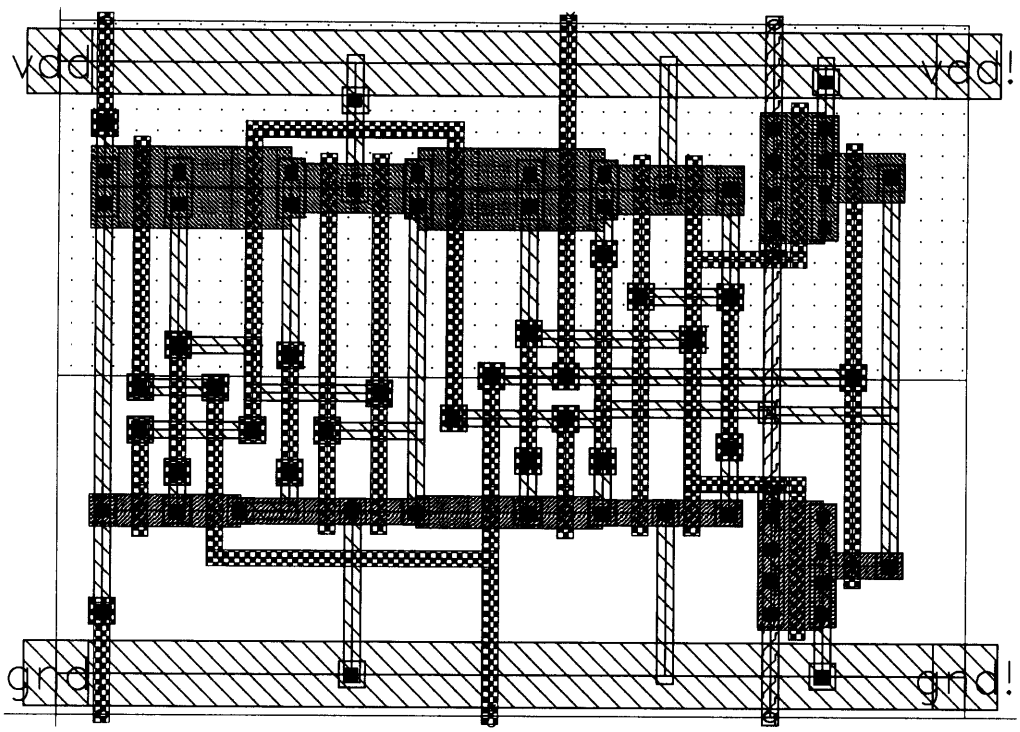


Figure A-4: D Filp Flop

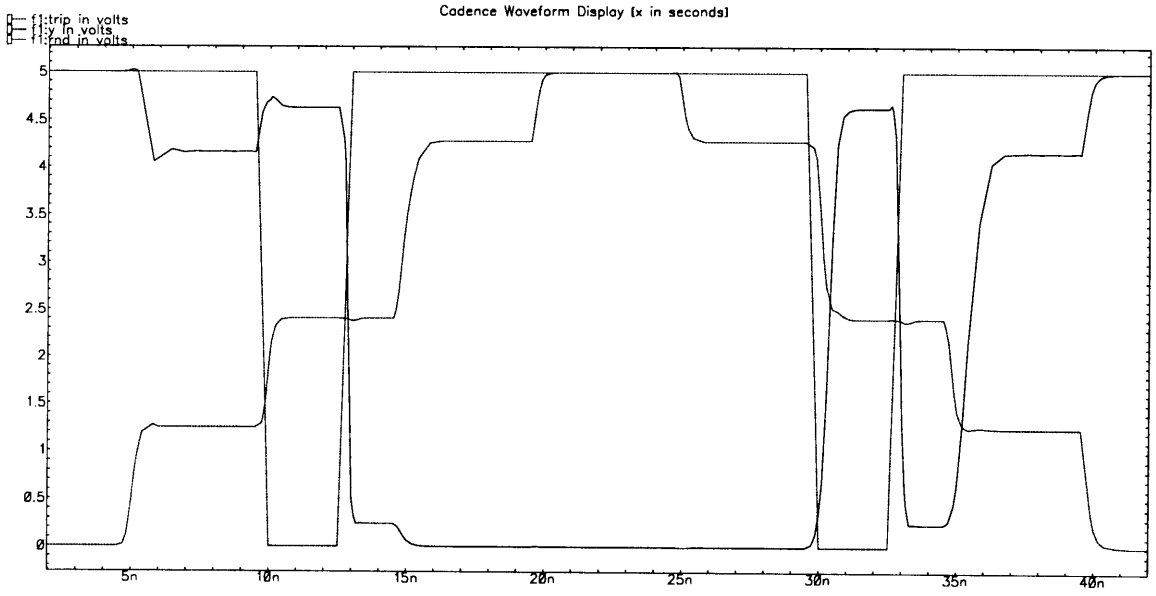


Figure A-5: HSPICE Simulation Results of Priority Comparator

Bibliography

[1] Arvind and R. A. Ianucci. Two Fundamental Issues in Multiprocessing. In *Proceedings of DFVLR Conference on Parallel Processing in Science and Engineering*, pages 61–88, West Germany, June 1987.

[2] V. E. Benes. In *Mathematical Theory of Connecting Networks and Telephone Traffic* Academic Press, New York.

[3] K. E. Batcher. Sorting Networks and Their Applications In em AFIPS Proceedings of the 1968 Spring Joint Computer Conference, Vol. 32, pages 307–314.

[4] Rishiyur S. Nikhil, Gregory M. Papadopoulous, and Arvind. *T: A Multithreaded Massively Parallel Architecture. In *Proceedings of the 19th International Symposium on Computer ARchitecture*. ACM, May 1992.

[5] Anant Agarwal, David Chaiken, Godfrey D'Souza, Kirk Johnson, David Kranz, John Kubiatowicz, Kiyoshi Kurihara, Beng-Hong Lim, Gino Maa, Dan Nussbaum, Mike Parkin, and Donald Yeung. The MIT Alewife Machine: A Large-Scale Distributed-Memory Multiprocessor. MIT/LCS/TM 454, MIT, 545 Technology Sq., Cambridge, MA 02139, November 1991.

[6] Arvind, D. E. Culler, and G. K Mass. Assessing the Benefits of Fine-Grain Parallelism in Dataflow Programs. *The International Journal of Soupercomputer Applications*, 2(3), November 1988.

[7] Anant Agarwal, Beng-Hong Lim, David Kranz, and John Kubiatowicz. APRIL: A Processor Architecture for Multiprocessing. In *Proceedings of the 17th International Symposium on Computer Architecture*, pages 104–114. IEEE, May 1990.

[8] M. J. Narasimha. The Batcher-banyan Self-routing Network: Universality and Simplification. In *IEEE Transactions on Communications*, Vol. 36, No. 10, pages 1175–1178, October 1988.

[9] Barbara A. Chappell, Terry I. Chappell, Stanley E. Shuster, Hermann M. Segmuller, James W. Allan, Robert L. Franch, and Phillip J. Restle. Fast CMOS ECL Receivers With 100-mV Worst-Case Sensitivity. In *IEEE Journal of Solid-State Circuits*, 23(1):59–67, February 1988.

[10] Frederic T. Chong. Performance Issues of Fault Tolerant Multistage Interconnection Networks. Master's thesis, MIT, 545 Technology Sq., Cambridge, MA 02139, May 1992.

[11] IEEE Standards Committee. *IEEE Standard Test Access Port and Boundary-Scan Architecture.* IEEE, 345 East 47th Street, New York, NY 10017-2394, July 1990. IEEE Std 1149.1-1990.

[12] Y.Tamir and G.L.Frazier. High-performance multi-queue buffers for VLSI communication switches. In *Proc. of 15th Annual International Symposium on Computer Architecture*, May 1988, pages 343–354.

[13] William J. Dally *et al.* The Message-Driven Processor: A Multicomputer Processing Node with Efficient Mechanisms. *IEEE Micro*, pages 23–39, April 1992.

[14] André DeHon. Robust, High-Speed Network Design for Large-Scale Multiprocessing. AI Technical Report 1445, MIT Artificial Intelligence Laboratory, September 1993.

[15] André DeHon. Fat-Tree Routing For Transit. AI Technical Report 1224, MIT Artificial Intelligence Laboratory, April 1990.

[16] André DeHon, Thomas F. Knight Jr., and Thomas Simon. Automatic Impedance Control. In *ISSCC Digest of Technical Papers*, pages 164–165. IEEE, February 1993.

[17] Mark G. Johnson and Edwin L. Hudson. A Variable Delay Line PLL for CPU-Coprocessor Synchronization. In *IEEE Journal of Solid-State Circuits*, 23(5):1218–1223, October 1988.

[18] Dake Liu and Christer Svensson. Power Consumption Estimation in CMOS VLSI Chips. In *IEEE Journal of Solid-State Circuits*, 29(6):663–670, June 1994.

[19] Kevin Lam, Larry R. Dennison and William J. Dally. Simultaneous Bidirectional Signalling for IC Systems. In *IEEE International Conference on Computer Design: VLSI in Computers and Processors* 1990.

[20] S. H. Unger and C. Tan. Clocking Schemes for High Speed Digital Systems. In *IEEE Transactions on Computers*, vol. 35, pages 880-895, 1986.

[21] H. B. Bakoglu. In *Circuits, Interconnections, and Packaging for VLSI*, pages 367-379, Addison Wesley, 1990.

[22] G. F. Pfister and V. A. Norton. "Hot spot" contention and combining in multistage inteconnection networks. In *IEEE Transactions on Computers.* C-34(10):943–948, October 1985.

[23] Charles E. Leiserson. Fat-Trees: Universal Networks for Hardware Efficient Supercomputing. In *IEEE Transactions on Computers*, C-34(10):892–901, October 1985.

[24] Tom Leighton and Bruce Maggs. Expanders Might Be Practical: Fast Algorithms for Routing Around Faults on Multibutterflies. In *IEEE 30th Annual Symposium on Foundations of Computer Science*, 1989.

[25] Tom Leighton and Bruce Maggs. Fast Algorithms for Routing Around Faults in Multibutterflies and Randomly-Wired Splitter Networks. In *IEEE Transactions on Computers*, 41(5):1–10, May 1992.

[26] Henry Q. Minsky. A Parallel Crossbar Routing Chip for a Shared Memory Multiprocessor. AI memo 1284, MIT Artificial Intelligence Laboratory, 545 Technology Sq., Cambridge, MA 02139, 1991.

[27] R. Smolley. Button Board, A New Technology Interconnect for 2 and 3 Dimensional Packaging. In *International Society for Hybrid Microelectronics Conference*, November 1985.

[28] E. Upfal. An $O(\log N)$ deterministic packet routing scheme. In *21st Annual ACM Symposium on Theory of Computing*, pages 241–250. ACM, May 1989.

[29] Kazumasa Suzuki *et al.* A 500 MHz, 32 bit, 0.4μm CMOS RISC Processor. In *IEEE Journal of Solid-State Circuits*, 29(12):1464–1473, December 1994.

[30] Frederic T. Chong Analog Techniques for Adaptive Routing on Interconnection Networks. Transit Note 14, MIT Artificial Intelligence Laboratory, April 1990.

[31] André DeHon and Frederic Chong and Matthew Becker and Eran Egozy and Henry Minsky and Samuel Peretz and Thomas F. Knight, Jr. METRO: A Router Architecture for High-Performance, Short-Haul Routing Networks. In *International Symposium on Computer Architecture*, April 1994.

[32] Thomas Simon and André DeHon. Transit Cannonical Form. Transit Note 55, MIT Artificial Intelligence Laboratory, November 1993.

[33] André DeHon. NOACT-ORBIT Data Sheet. Transit Note 91, MIT Artificial Intelligence Laboratory, September 1993.

[34] André DeHon. NIACT-ORBIT Data Sheet. Transit Note 92, MIT Artificial Intelligence Laboratory, September 1993.