Implementation Issues for a Packet Switched Routing Chip

by

Gowri A. Rao

S.B., Massachusetts Institute of Technology (1993)

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

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1994

Author .... [illegible]
Department of Electrical Engineering and Computer Science
May 16, 1994

Certified by .... [illegible]
Dr. George A. Boughton
Research Associate
pervisor

Accepted by .... [illegible]
F. R. Morgenthaler
Chairman, Departmental Committee on Undergraduate Theses

Implementation Issues for a Packet Switched Routing Chip

by

Gowri A. Rao

Submitted to the Department of Electrical Engineering and Computer Science
on May 16, 1994, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The purpose of this project was twofold. My first objective was to study the transmission and timing characteristics of the input pad section of ARCTIC, a packet-switched routing chip. ARCTIC (A Routing Chip That Is Cool) will be used to construct the interconnection network for *T (StarT), a dataflow computer architecure. In addition, my goal was to apply this research on a larger scale by designing a smaller version of a packet switched routing chip. In this way, I was able to meet my goal of gaining a broader understanding of packet routing for multiprocessor networks.

This research will depend heavily on the work done in my Bachelor's thesis concerning chip-to-chip communication and the core circuitry for the proper latching of data in the input section of ARCTIC. This Master of Engineering thesis will be performed under the guidance of Dr. G. A. Boughton of the Computation Structures Group of the Laboratory for Computer Science.

Thesis Supervisor: Dr. George A. Boughton
Title: Research Associate

Acknowledgements

I would like to thank my thesis advisor, Dr. G.A. Boughton for all of his help and guidance over the past two years. Without him, this thesis would not have been possible. Also, many thanks to Prof. Papadopoulos for his advice over the past four years and to Profs. Arvind, Papadopoulos and Dr. Boughton for their generosity in supporting me for my Master's degree.

I would also like to thank Roberta L. Bailey for her part in our 6.371 VLSI Design Project, and Satoshi Asari, who worked with me in 1992 when we began working in the Computation Structures Group as UROPs.

Of course, I could not have made it this far without the love and support of my family. I thank them for being there.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

This thesis for the Master of Engineering degree was done as an extension of my previous bachelor's thesis, "Transmission and Timing Issues for ARCTIC Transmit and Receive Apparatus[2]." In my work towards my S.B., I began by focusing on the transmission issues surrounding the chip-to-chip interconnection of ARCTIC, a packet router for a high performance multiprocessor network. By modeling the system using HSPICE and Verilog HDL, I was able to bring to light some of the transmission issues for high frequency interconnections.

As I moved along in my research, I then began to study the digital timing aspects of the input and output pads of ARCTIC as well. This work resulted in the preliminary design of a delay line that was needed in the input pads of the system [2].

My goal in this thesis was to gain a broader understanding of multiprocessor routing issues by expanding on my previous work, as well as exploring other methods of routing. I relied heavily on PaRC, a precursor to ARCTIC, as well as on some approaches to routing that were entirely different. By studying C. Joerg's paper [1], I was able to enhance my knowledge of ARCTIC as well as to broaden my outlook on interconnection networks in general.

To bring all of this together, I decided to design and implement my own routing chip [3]. I began my design as a joint project with Roberta Bailey, for the Introduction to VLSI Systems course. The plan for the chip was carefully thought out, implementing features that presented themselves through research papers on routers. The chip was then designed using Mentor Graphics GDT layout tools and VHDL synthesis tools. Unfortunately, the chip was not able to be fabricated.

The design for this CAM based router chip maintained only a few superficial similarities to the ARCTIC endeavor, in that it is a packet-switched router to be used in a multiprocessor interconnection network. However, the two designs differ greatly from there. For example, in my design, I decided to implement a Content Addressable Memory to store routing information. This feature is not present in ARCTIC. In this discussion, I attempt to present some background material on ARCTIC and PaRC, it's predecessor, as well as to explain the design and implementation issues that came up in the design of my own packet routing chip.

1.2 Background on ARCTIC

The Computation Structures Group of the Laboratory for Computer Science is studying the *T (Star T) system as an alternative to the previous Tag Token and Monsoon data flow computers. While these systems relied heavily on exploiting instruction level parallelism, the goal of the *T system lies more with a strategy called multithreading in addition to making more efficient use of the hardware in the machine [16].

The *T is a 16-site structure, where each site contains four PowerPC data processors, a memory system, and a Network Endpoint System (NES). The NES consists of a PowerPC network processor, a shared memory unit, and a Network Interface Unit (NIU). My work involved the router's interactions with the NIU [5].

The interconnection network for the *T system will be constructed using ARCTIC (A Routing Chip That Is Cool), a four input, four output packet switching routing chip. Although most of ARCTIC will operate using a single 50 MHz local clock, each of the input sections will require a separate 50 MHz clock. These input clocks are required to maintain a frequency extremely close to the local clock, however, their phase may vary [5].

The data links that connect any two ARCTIC chips consist of a 16 bit wide data path,two clock signals and a frame bit. In the input unit of the chip, the 16 bit wide data path will be demultiplexed into a 32 bit wide path for use within the digital circuitry of the chip. The clock signal is transmitted to the chip in two phases which are 180 degrees out of phase. For this reason, we need two clock bits included in the data link. See Section A.1 for more information about the clocking scheme as it pertains to the timing issues of this project. In addition, a frame signal is sent with the data. This signal, which is Manchester encoded, is used in determining the frame of a packet of data. See [5] for a more detailed description of the ARCTIC environment.

For the purposes of this project, I will be concerned only with the data link signals as they pertain to the timing and transmission issues between two ARCTIC chips.

1.3 Background on CAM based packet router

The Content Addressable Memory (CAM) provides a natural solution to any search or comparison problem. Similar to a regular memory cell, each CAM cell has one-bit storage capability. However, the CAM cell has an additional comparator, implemented in hardware, which will indicate whether any given data matches the stored data in the cell. In this way, an array of such cells can be addressed by their contents, rather than by cell number or index [14].

The first widely used application of CAMs was in virtual memory management and cache subsystems for CPUs [11]. A special purpose CAM is a Translation Lookaside Buffer (TLB), which is used for quick address translation in a virtual memory system.

CAMs soon became recognized as the solution to yet another class of problems involving communication systems. The Am99c10, a commercial CAM, was designed for use in LAN bridges in communication systems, where a method was needed to make a very fast comparison against a list. Because of the speed of CAM arrays, the CAM was found as a workable solution for destination address-based message forwarding in FDDI LAN bridges [12].

Furthermore, researchers at Columbia University found that CAMs were a very cost effective way to implement routing tables for communication networks [13]. For tables with a small number of entries and a large address space, it was shown that CAMs were an optimal solution.

After assessing the results of these studies, I decided to explore a CAM based approach in the design of my own router. The CAM was used not only for performance, but as a challenging exercise in circuit design as well.

Chapter 2

Background: the PaRC network

The precursor to the ARCTIC chip was PaRC, a packet routing chip designed by Christopher Joerg. In his paper, "Design and Implementation of a Packet Routing Chip," [1] Joerg presents many of the issues surrounding router design for multiprocessor networks. PaRC is designed specifically for the multiprocessor interconnection network for the Monsoon parallel computer. Monsoon is a parallel processing dataflow computer, which executes the data-driven model of computation. A dataflow processor operates by receiving and transmitting packets of data over its interconnection network. Because of the many processor nodes which must interact, a high bandwidth, low latency network router was needed. PaRC was designed with this in mind.

In this section, I will examine the issues involved in network design for multiprocessors by reviewing Joerg's paper on PaRC. I will present some of the architectural features of PaRC, and compare PaRC's features with other options where possible. I found that a good source of information on networks for comparison can be found in "Robust High-Speed Network Design for Large Scale Multiprocessing," [4] by André DeHon. DeHon presents a good overview of networks in the first few chapters of his paper, and I have used some of his ideas in the areas of Topology, Routing Schemes, and Fault Management.

2.1 Network Topology

PaRC is a 4 input, 4 output packet switch on a chip. Although it was designed specifically for Monsoon, it can potentially be used in several different networks.

2.1.1 Butterfly Networks

Monsoon's network is an n-cube butterfly topology. The butterfly network can be implemented with switches connected in a crossing manner. One advantage of the butterfly network is that the same size nodes can form any size network. The complexity of the node does not have to increase with network size. By comparison, in a case such as the hypercube network, the complexity of each node varies with N, the number of nodes in the network. In addition, when cascaded in a recursive structure, a properly constructed butterfly network can exhibit behavior similar to a crossbar, in which every node has access to every other node in network [4]. However, one problem that is inherent in the structure of a butterfly network is that some patterns of interconnection are not available. This could impose limitations on fault tolerance, which will be discussed in a later section.

A message passing through a butterfly network goes through O(logN) switches on its way to its destination. This is an advantage, when compared with a 2D mesh topology, in which it has to pass through $O(\sqrt{N})$ switches. However, if we consider a 3D mesh topology, the number of switches reduces slightly to $O(\sqrt[3]{N})$ switches. In most reasonable sized networks, where $N < 5000$, $O(\sqrt[3]{N})$ is comparable to O(logN). In this case, the mesh network has the advantage of locality over the butterfly. However, for the purposes of Monsoon, locality is not exploited, so the butterfly network is sufficient [1].

2.1.2 Direct and Indirect Networks

Another feature of the butterfly network is that it allows an indirect network, as used in Monsoon. In a direct network, each switching element is connected to a processor node. In this situation, one can take advantage of processor locality in designing the exact topology of the network. By writing programs in such a way that messages travel most frequently to the nearest nodes, we can minimize the average distance traveled by messages over the entire network. In this way, we can attempt to optimize the speed of the network. In an indirect network, however, only switches in the first and last stages will be associated with nodes. In this type of network, routing can be optimized as in any network, however the processor destinations will only be in the first and last stages. In a dataflow architecture such as Monsoon, a direct network is not needed because there is no sense of locality between processors. When one node needs to send a message, it is equally likely that the message will go to any one of the other processors [1].

2.1.3 Fat Tree Networks

The PaRC router can be used effectively in a fat tree network, as well. A fat tree has the basic shape of a classical binary tree, with nodes at each leaf of the tree. Each node contains a switch, so that messages may be passed either up or down the tree structure. A given packet will first move up the tree as far as it needs to go, then down to its destination. Each switch therefore effectively chooses between two directions. The fat tree gets its name due to the fact that as we move up the tree structure, the number of links connecting two nodes doubles at each level of the tree. In other words, each level of the tree has twice the bandwidth of the next lowest level. This structure can be seen in Figure 2-1. Most importantly, though, the fat tree network is one which allows us to take advantage of locality between processors [4]. Nodes which are nearest neighbors are much more easily accessible than remote nodes.

2.2 Routing Scheme

The PaRC chip supports fixed size packets of 12 words each. These packets consist of a header word, which contains the network destination address for the packet, 9 data words for the processor, and 2 CRC words for error detection. The destination

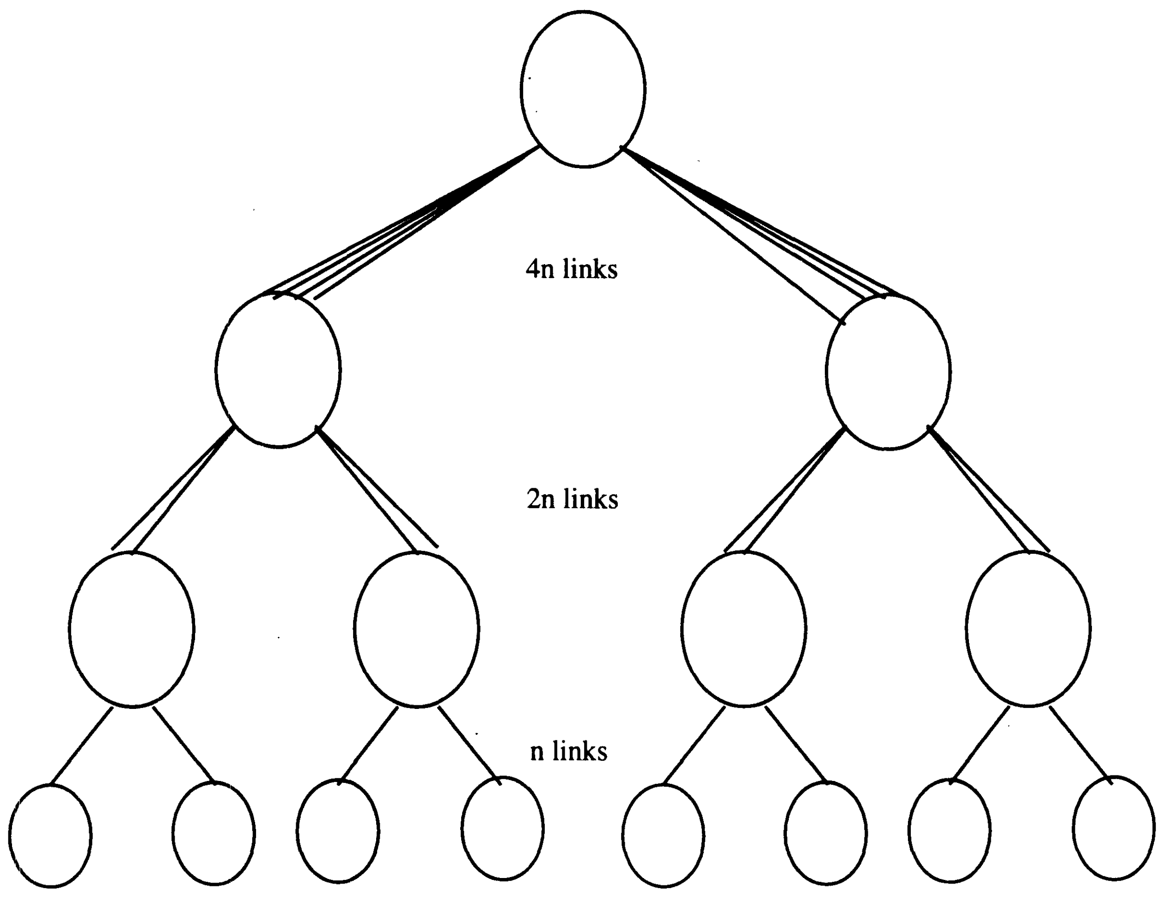

Figure 2-1: Illustration of a Fat Tree

address is then used in a self routing scheme to determine the output port through which the packet should be routed.

When a packet first comes in from the network, it is stored in the input port while it waits to be routed. Each of PaRC's 4 input ports has buffer space for 4 packets. The input port is also responsible for checking the incoming packet for errors. This will be discussed in more detail in the Fault Tolerance part of this discussion. When an input port is full, a flag is generated which deters other nodes from sending packets to that port until further notice. The sender would then try to resend it either to a different node, or to the same node on a later cycle. This way maximum usage of the buffer space can be achieved. If a packet's path is clear, it will exit the switching element through one of the 4 output ports. Each output port has a scheduler and a transmitter to control the routing of packets. The transmitter must handle receiving and transmitting the acknowledge signal in the case of a circuit switched packet. Circuit switching will be discussed in more detail in Section 2.2.2.

2.2.1 Buffer Utilization

One issue which can greatly affect the performance of a network is buffer utilization [1]. One way to go about routing would be to have one long FIFO buffer to store incoming packets. This would save space on the chip, as well as ensure that the packets left the router in the proper order. However, with one long fifo, if the topmost packet is blocked on a given cycle, then all of the packets behind it must wait to be transmitted. For this reason, PaRC was implemented with 4 separate buffers per input port. This was meant to increase the throughput of the system in the following way: if the first packet is blocked, the scheduler can go down the list in order until it finds an unblocked packet to send. In addition, with this scheme, multiple packets can be sent out on the same cycle. If all packets are unblocked, then they can all be sent simultaneously, greatly reducing the waiting time compared with the fifo case. However, this obviously has a big cost in hardware, because of the increase in switching complexity. With PaRC's scheme, a 16 x 4 crossbar capability is needed, as compared with only 4 x 4 crossbar capability in the FIFO case.

2.2.2 Circuit Switching vs. Packet Switching

PaRC has two main switching methods to consider: packet switching and circuit switching. In the circuit switching model, a packet would hold onto all the links which it crossed from the sender through the receiver. This means that all of these links would have to be available for the packet to make it to the destination. In the case where one of the links was down or busy, the scheduler would have to try and resend the packet until all of the switching nodes needed were available. When the message arrives, there is a full connection from the sender to the receiver. This allows for the receiver to send an acknowledgment signal back through all of the nodes to the sender.

In packet switching, no acknowledgment signal is needed from the receiver to the sender. The packet simply travels as far through the network as it can, and then waits at each step until transmission can be completed. This is different from the circuit switching case in that a packet can be sent only part way to its destination. In circuit switching, the links used for transmission are not free until the packet gets all the way through. Therefore, in packet switching as packets travel from link to link, when a packet leaves, the link is freed to route another packet.

For the PaRC chip, packet switching was chosen as the main method of routing. However, in certain cases, circuit switching may be needed to ensure that a packet has reached its destination. One such case would be during memory deallocation. Before an area of memory can be deallocated, the source needs to acknowledge that the last write to that area has been completed. For cases such as these, PaRC allows for a circuit switching option. The method of switching is therefore signified in one bit of the packet format. For packet switching, PaRC is in normal operation. For circuit switched packets, the transmitter at each stage will have to keep a back pointer to the previous switch. PaRC was designed in such a way that normal packets can go through a switch even when it is waiting for an acknowledge from a previous send. The acknowledge signal travels separately once the packet is received. This way, all normal packets are not blocked when a circuit switched packet is trying to get through the network.

2.2.3 Buffer Scheduling

When considering how to schedule the packets through the nodes, PaRC considered two distinct methods: centralized scheduling and distributed scheduling. In centralized scheduling, a chip has one global scheduler for all ports on chip. The scheduler can therefore "see" all of the packets waiting to be sent out and optimize its strategy for greatest efficiency. In distributed scheduling, each output port has its own scheduler. In this method, there is a potential problem of packets getting out of order, since each port can only "see" the packets which it schedules. To ensure the proper ordering of packets, PaRC assigns a priority value to each packet. At each port, there is a fifo buffer which is used to store pointers to packets, according to their order of priority. This way, the next packet to be sent can easily be determined by simply reading off the top of the fifo.

As mentioned before, PaRC has 4 schedulers on the chip, one per output port. The basic operation of packet transmission is as follows: any switch will accept packets as long as it has room to buffer them. Therefore, if the destination for a given packet has all four of its buffers full, it will generate a wait signal and the packet will be blocked. Packets will be sent to their specified output port unless they are blocked.

2.3 Fault Tolerance

2.3.1 Network Fault Tolerance

In order for a system to be robust, it must be able to detect when faults have occurred and handle these faults in an appropriate manner. Faults are generally detected by passing some form of redundant information along with the transmitted data. This information, usually in the form of a checksum or parity bit can then be compared with the data each time it is received at a new port. In this way, we can detect when a packet has been sent incorrectly.

The next issue of concern is how to handle a fault after it has been detected. The most simple schemes will simply discard bad data, and perhaps generate an error flag of some sort. In more complex systems, it is often necessary to analyze the cause of the fault and generate signals to correct it in order to ensure proper coherence in the system [4].

2.3.2 Error Checking in PaRC

The error checking mechanisms in PaRC are designed to detect virtually all link transmission errors. The first type of error which PaRC can detect is a room error. As mentioned before, each input port has room to buffer only 4 packets. In normal operation of the system, there is a flow control mechanism which prevents overflow of a buffer. However, in the case where a packet gets through and there is no room to buffer it, the room error is generated and the misdirected packet is discarded.

PaRC is also able to detect CRC errors. As a fault tolerance mechanism, the CRC checksum for each packet is included as 2 words which follow the data words. The input port has circuitry which accumulates a checksum while the packet is being received, and a CRC error occurs if the CRC code at the end of the packet does not equal the checksum calculated by the input port.

Normally, when packets are not being received, a designated idle pattern is transmitted. An idle error occurs if a switch receives a bogus word which is neither part of a packet, nor the idle pattern. In summary, PaRC is able to detect almost any transmission error that could occur. Any error inside a packet can be detected with a CRC error. Any error which occurs during idle time can be taken care of with the idle error. And overflow occurrences in the input buffers will be caught with a room error.

2.4 Analysis

2.4.1 Design Efficiency

In the first part of this discussion, I looked at the differences between direct and indirect networks. PaRC relies on an indirect network because processor locality can not be optimized for Monsoon. This choice greatly simplifies the network and is an efficient choice for PaRC's given application. However, if we consider using PaRC in other networks as well, a direct network would definitely be more useful in certain applications.

Next there was the issue of packet switching and circuit switching. PaRC chose to mainly implement packet switching because for most data transmissions, acknowledgment is not needed from the receiver to the original sender. When we consider that in packet switching, a blocked packet can get part way through the network before having to wait, it is clear that this allows for more throughput in the system. Although Joerg does not offer any experimental evidence that his potential for throughput gets exploited, I believe that the choice does indeed result in more efficient use of the bandwidth of the system. Of course, this largely relies on the effectiveness of the scheduling strategy that is used. However, in this scheme a packet may actually suffer in latency. In a circuit switched packet, latency could be introduced in the establishing of the path. However, nodes are usually less complex in circuit switching than in packet switching. Therefore, circuit switching will usually result in less latency compared with the packet switched case [1]. This latency can be minimized by increasing the number of paths to the destination. By considering different network topologies, the packet latency can be optimized.

The issue of centralized vs. distributed scheduling is clearly a tradeoff between time and space. The separate output circuitry which is needed in the distributed case is balanced out by the potential slowness of having one big scheduler for the entire chip. However, it is unclear whether the interactions between the four separate schedulers and output circuitry adds to the delay in the distributed method.

One area in which PaRC paid a lot of attention is in buffer utilization. In his analysis of using separate buffers as opposed to one long fifo, Joerg cites a best case increase in utilization from 67 % to 80 %. Because PaRC looks at all four packets in each port, the best case utilization rises to 90%. Finally, the addition of separate output circuitry to each buffer lends the ability of each buffer to be read independently. Joerg cites a best case link utilization rate of 99 %. Because of the design constraints imposed by Monsoon on this router chip, I believe that one of the few things that PaRC was entirely free to optimize was the buffer utilization. This problem was solved effectively, since PaRC can only support fixed sized packets. Obviously, the buffering strategy would have to change significantly if variable length packets were to be supported.

2.4.2 Fault Management

Finally in the issue of Fault Management, I think that PaRC was very thorough in its attention towards detecting faults. One important system feature that Joerg described was that in Monsoon, the sending processor assumes that its packet was received at the destination. Because of this, and to decrease the complexity of the chip, it is probably sufficient that PaRC only marks error detection in a packet, rather than addressing error handling to recover from an error. This was based on the assumption that the network is reliable, and no errors would occur.

In general, however, lost packets could easily break down the proper operation of the system. Therefore, in the case of a less reliable network where we expect more errors, a more complicated scheme would be necessary to recover corrupted data and retransmit it.

In conclusion, Joerg's PaRC project was a good example in the issues involved in the design of a router for multiprocessing. By using André DeHon's paper as a reference, I was able to compare Joerg's approach with other possibilities presented in the first few chapter's of DeHon's paper. In designing the CAM based router, I looked at many of the issues which Joerg outlines in his paper, and used his paper as a guide in the design and implementation of my packet router chip.

Chapter 3

A Packet Switched Router using CAM

This project involved the design and implementation of a packet routing switch which uses a content addressable memory (CAM). The goal was to construct a versatile, general purpose router that demonstrates the functional features of modern routers used in multiprocessing. In order to maintain both simplicity and functionality, I have designed a two input, two output router which can be used in several network topologies. Although the naming conventions of the actual ports (North/East/West/South) imply a two dimensional mesh array, this router can also be used in a butterfly network or a Banyan network, for example.

The router will support a network of up to 16 switching nodes, using a 4 bit address to designate them. Depending on the application, these can be used in conjunction with a 16 processor node network, or as bridges between several different parts of a larger network. The CAM has only eight locations, but it can handle a larger number of active nodes by making use of the default direction.

This router was designed jointly with Roberta Bailey as part of a 6.371 VLSI Final Project. Roberta designed the input fifo, and three of the control FSMs. I will concentrate mainly on the sections which I designed: the CAM/RAM block, the address multiplexor, and the control system for CAM address loading.

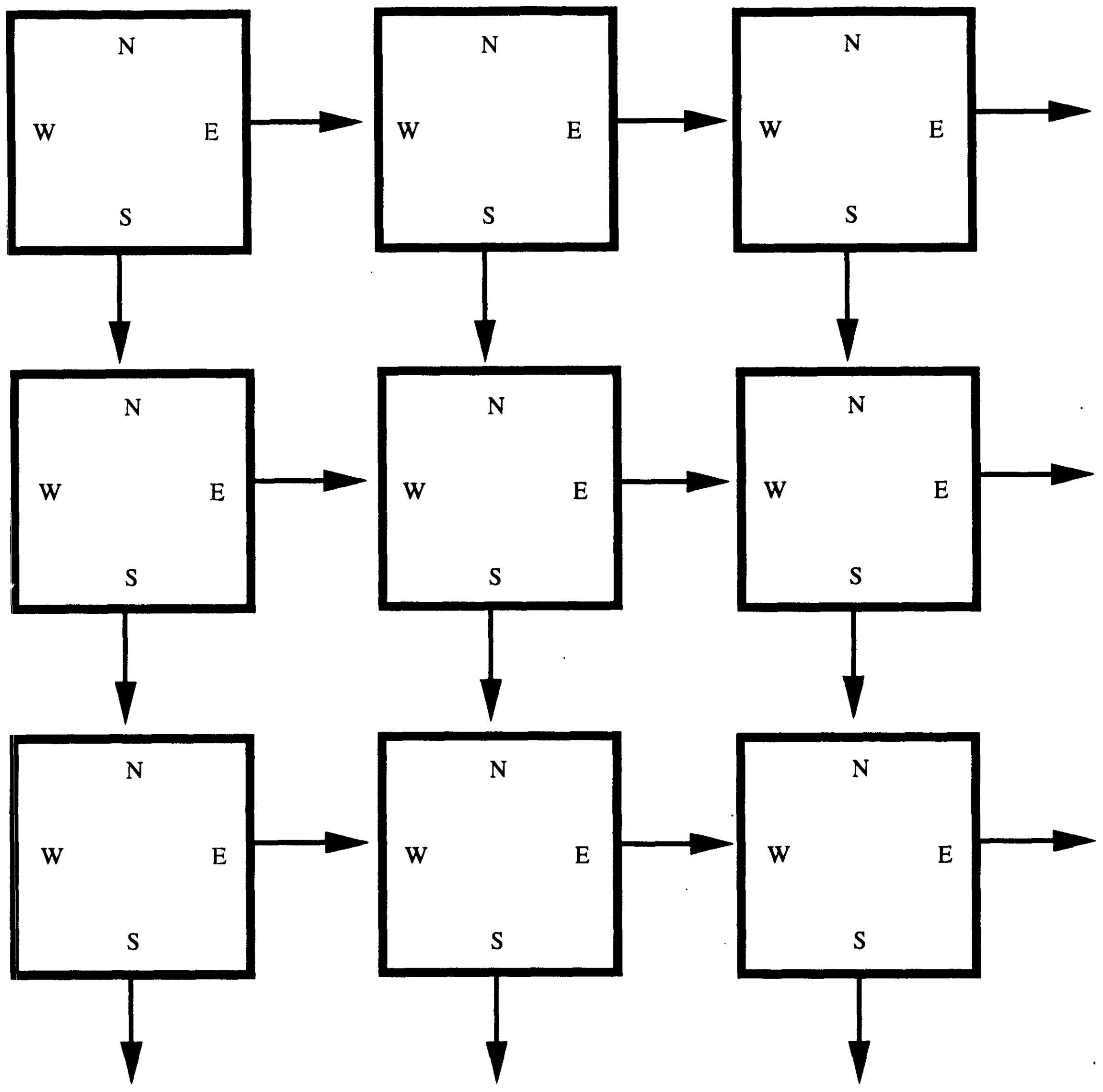

Figure 3-1: A 2-Dimensional Mesh Network of Routers

3.1 Overview

The main function of the router is to route fixed size packets from its two inputs (West and South) to its two outputs (North and East). An example of the 2D mesh network is shown in Figure 3-1. The packets will be received through a 4 bit wide data path into one of the two input FIFO buffers, as shown in Figure 3-2. The destination address for the packet will then be read into an address multiplexor. In the normal operation of the chip, the 4 bit address from one of the FIFO buffers will be fed to the CAM, which will compare it to its internally stored addresses. If a match occurs, the resulting direction will be output through the direction RAM, and latched for use by the output multiplexor. The output mux will then wait for an acknowledgment signal through the appropriate output port and then transmit the full packet through that port. If a match does not occur the packet will be routed through the default path stored in location 0 of the CAM. In addition an address error will be generated.

During normal operation, new address and direction information can be loaded into the CAM during the idle time while a packet is being routed through the output mux. This new information is loaded in from the network through a 2 bit datapath into a buffer, where it will be held until it can be loaded. In addition, the router has a CAM load mode where inputs are disabled and the CAM can be continuously loaded at a faster rate.

There is also a time limit on the amount of time that the output mux will wait for an acknowledgment signal from the destination router in order to route the current packet. Once this time is exceeded, the packet from the other FIFO buffer will be enabled and an attempt will be made to route it instead.

3.2 Address Storage and Evaluation

The routing function of this chip is to receive an address off of an incoming packet, compare it with an internal list, and determine the destination of that packet. The router is therefore required to have hardware to support both storage and evaluation

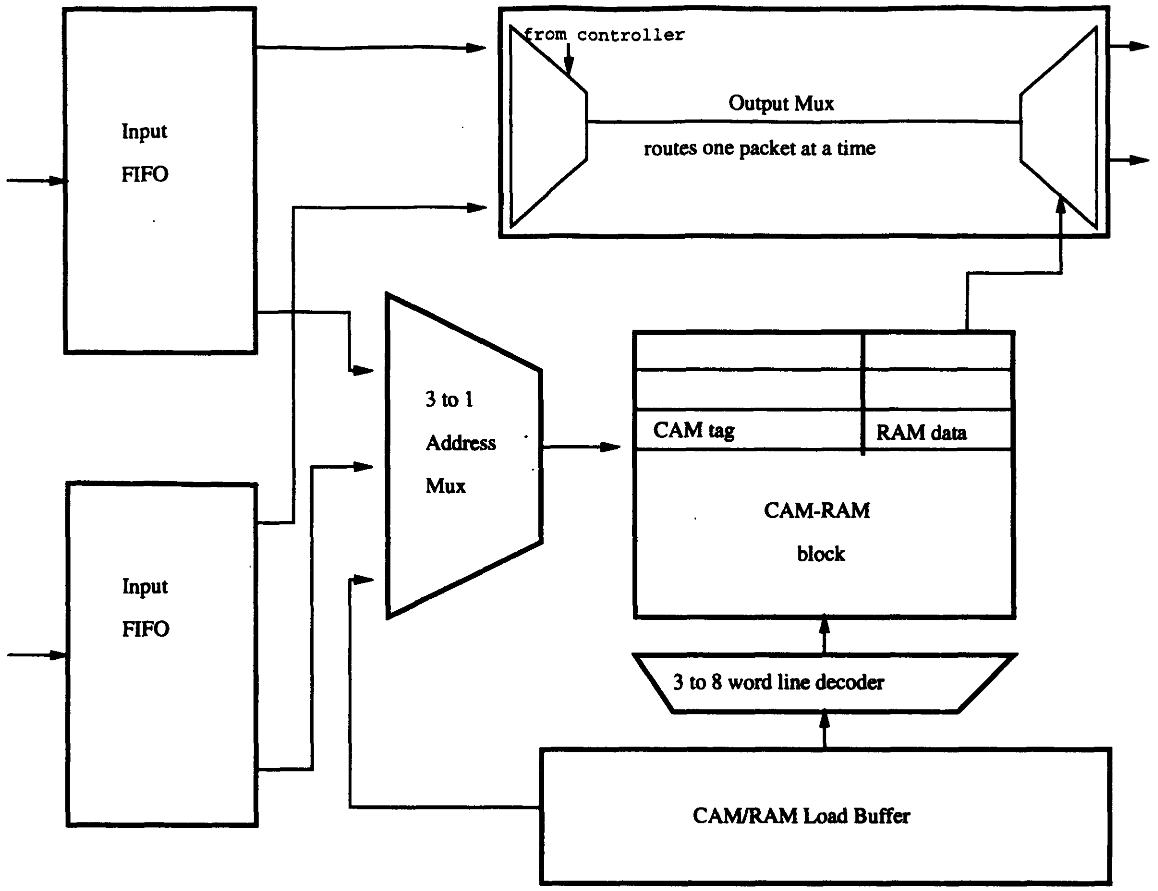

Figure 3-2: Data Path for the CAM based Router

of network addresses. The storage of addresses is accomplished by the CAM, while the RAM and the Output mux are used to evaluate the direction for the given addresses. This is shown in Figure 3-2.

3.2.1 The CAM-RAM Unit

The Content Addressable Memory block stores the 4 bit address data for each destination in the network. In this case the CAM is a 4 x 8 block, to accommodate 8 of the possible 16 router locations in our network. It takes as inputs the 4 bit destination address for each packet, which it receives from the Address Mux. It then compares this data with the contents of each CAM cell and asserts a match signal which corresponds to the correct location. The CAM provides match lines directly to the word lines of the RAM, which then outputs the direction information for the Output Mux. In addition to the match lines, the CAM will output a MATCH signal, indicating that a match has been found. This will be used to generate the both the address error signal on a mismatch, and the dir_rdy signal which tells the route mux that the direction bit is ready to be read. The RAM is used to store the 1 bit direction data (North or East) for the destination node addressed in the CAM. It will receive the match lines from the CAM at its read inputs, and latch the corresponding direction bit into a static register for the Route Mux to access. A static register is needed to ensure that the direction bit will remain valid throughout the entire packet transfer, until the SWAP signal from the output FSM.

A block diagram of the CAM-RAM unit can be found in Figure 3-3. In this diagram, the address supplied on the left is simultaneously compared with the contents of each CAM location. If a match is found, the direction from the corresponding RAM cell will be supplied to the output mux. In this example, if we were to apply the network address of 1011 to the CAM, a match would occur at the first CAM location. The corresponding output direction of 1 (East) would then be supplied to the output mux. In this way, the CAM-RAM block acts as a fully associative memory.

3.2.2 CAM/RAM Load Buffer

New routing information can be loaded into the CAM and the RAM either during idle time between packets or while a packet is being transmitted but after the CAM has finished evaluating the address and set the Direction Ready signal. The CAM/RAM Load Buffer will be a serial in/parallel out buffer which stores new routing information received through 2 serial data input pins. In a manner similar to the FIFO described below, it will use a shift register to successively load in 8 bits of data: 4 bits of new CAM address data, 1 bit of new direction data and the 3 bit location which will be decoded to select a line of the CAM/RAM block to be replaced.

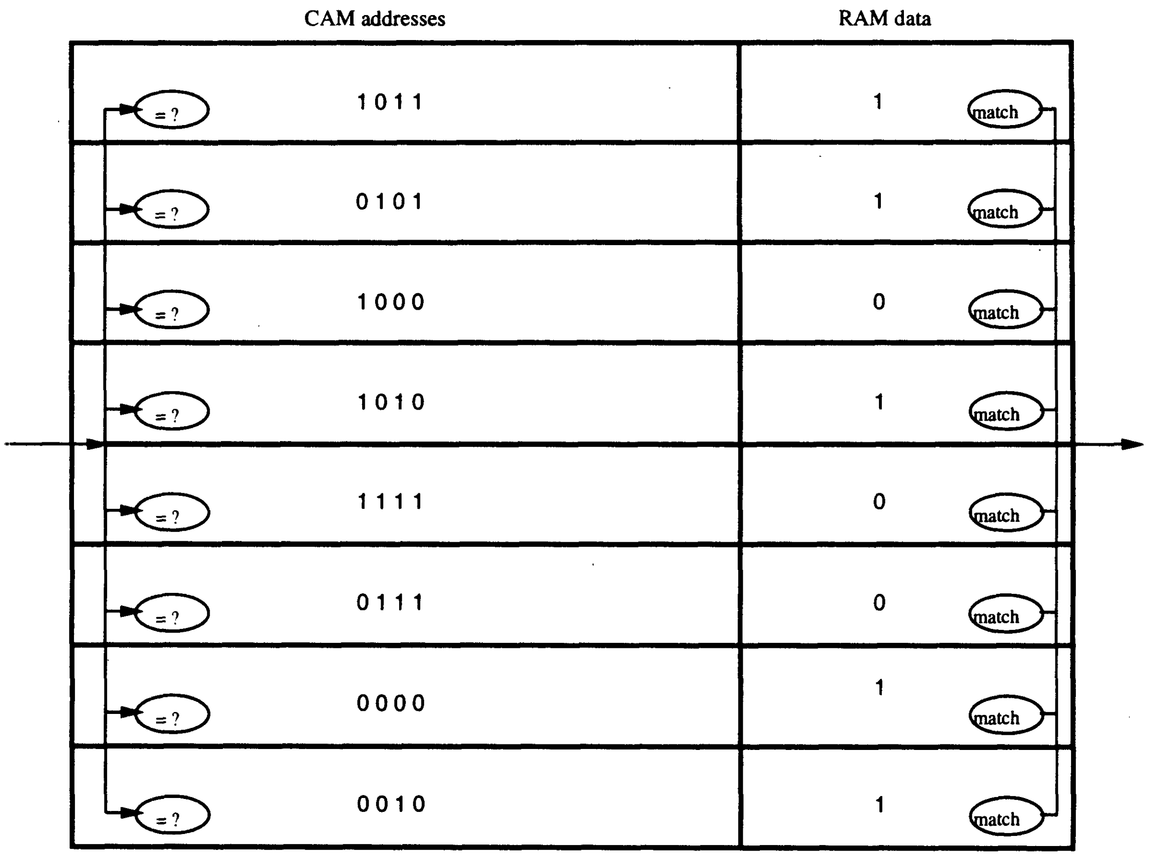

Figure 3-3: The CAM-RAM Unit

3.2.3 Address Multiplexor

The address mux is used to select a four bit address to present to the CAM. In normal operating mode, this address will come from one of the two input fifos based on a priority signal from the address FSM. In the case of a CAM address reload, the address mux will select the four bit address from the CAM reload buffer, and present it to the CAM, while the new data is presented on the CAM's data lines. Once a packet transmission is set up, it can load a new CAM address without interrupting the transmission in progress.

3.3 Packet Storage and Transfer

Data packets are initially stored in a FIFO buffer until the address has been decoded and the destination has acknowledged that it is ready to receive the packet.

3.3.1 FIFO Buffer

There are two identical fifo buffers on chip, one at each input port. Each fifo is large enough to hold exactly one fixed length packet. The packet is loaded in 4 bits at a time through one of the two input ports, and is held in the fifo until the routing direction is determined. At this time, it is transferred to one of the output ports via the output multiplexor.

3.3.2 FIFO Decoder/Shift Register

A row decoder is needed to enable loading and reading of the fifo buffers. This is implemented by a resettable shift register which controls the loading of the fifo, 4 bits at a time. The shift register also recognizes when the buffer is full, and signals to the control FSM.

3.3.3 Output Multiplexor

The output mux sets a path between one of the inputs and either one of the outputs. The path is based on the source priority from the address FSM and the destination direction from the direction RAM. The controls and path are latched in and signalled by the direction ready flag (dir_rdy) and are maintained until the next packet is selected. This mux provides a path between an input fifo to the appropriate output port for 4 data bits at a time. It also directs the handshaking control bits to the receiver and control signals between the output FSM and the selected input fifo.

3.4 Control Design

In order to simplify and reduce the size of the control for this chip, the control functions were split into multiple smaller FSMs. This had the added benefit of allowing independent operation of the two input fifos. Each Control block was individually modeled in VHDL at the behavioral level and then several were connected and tested in combination. The state equations and gate level schematic were done by hand and entered into Led using the standard cells library. This was then synthesized into a layout using the AutoCells procedure.

3.4.1 Control Subsystems

Input FSM

Each input fifo and its external control bits are independently controlled by an input fifo FSM. This allows data to be loaded into one fifo while the other is loading, waiting or transmitting. The inputs are the external DAV (Data AVailable) signal, the Fifo Full flag, and the Start Of Packet (SOP) signal from the fifo as well as the global mode signals. The outputs are the external acknowledge signal (ACK) and an inc_fwr (Increment/Reset) signal to the fifo write decoder It has four states stored in two state bits.

Address FSM

The address FSM is the most complex controller of the chip as it coordinates all of the other controllers. It prioritizes the address requests of the two fifos (FA_RDY and FB_RDY), and the Cam load data (CA_RDY) sends the selected address to the CAM with a LD_CAM or EVAL_ADD signal to indicate whether an address is to be loaded or evaluated. Upon receipt of a DIR_RDY, it checks for new CAM data to be loaded while data is being transmitted. Upon a LD_DONE or DIR_RDY from the CAM, it waits for a command from the output FSM to check for the next fifo address. It also maintains the current source direction in a state variable to select the output mux direction and to ensure even data transfer between the inputs.

Output FSM

The output FSM triggers the fifo transmission, communicates with the receiver and signals the address FSM to swap priority between inputs. The inputs to the FSM are the external ACK signal from the receiver, the DIR_RDY signal from the CAM/RAM, the EOD (End Of Data) signal from the fifo and a time-out signal from a local counter. The outputs are an external DAV (Data AVailable), a trigger for the fifo read decoder to initiate FIFO transmission, a swap signal to the address FSM for the next priority if the packet is complete or the wait timed out and an increment signal for the local time-out counter.

Load FSM

The loading of the CAM reload buffer is controlled by this FSM. It takes as an input the CAM Address Available signal from the network, which signifies to the FSM to begin loading serial data through the two serial data input pins. It then provides control signals for loading successive locations of the CAM reload buffer. Once the buffer is full, the Load FSM will indicate that the address is ready to be loaded with a handshaking signal. The address FSM will then return the handshake and enable the buffer to simultaneously apply its outputs to the CAM and the RAM. At completion of the CAM Load, the Load FSM will return control to the Address FSM, and normal operation can be resumed. There will be no signal to prevent the buffer from loading data from the network. Therefore it is up to the external system to ensure that the CAM reload buffer loads only once per packet period in normal operating mode, or three to four times per packet period when the CAM load mode has disabled the packet inputs. This limit of three to four loads per packet period was imposed to ensure that data in the CAM reload buffer does not get overwritten before it has been successfully loaded into the CAM.

3.4.2 Network Protocol

External I/O Control

In order to ensure proper interactions with the surrounding system, we must define a handshaking protocol to facilitate external I/O control. The router chip therefore has two control lines which are used for handshaking between the sender and receiver.

The Data Available signal (DAV) from the sender is high when data is waiting to be sent and stays high during transmission. It goes low according to the sender's internal End Of Data (EOD) signal or if the sender's Output FSM times out waiting for the receiver to acknowledge.

The Acknowledge Ready for Data (ACK) signal from the receiver is high when the FIFO is ready to accept data. This is used to trigger data output from the sender. This goes low when the receiver's FIFO Full flag is tripped.

In addition, a Start of Packet (SOP) signal is used to designate the first bit of data in a packet. The one bit signal is located in bit 0 of the first nibble in each packet. It is recognized by the receiving chip either immediately following a previous packet, or after the idle pattern.

CAM Address Loading

In normal operation, CAM reload data is limited to one new address per packet period to allow it to interleave with data packet transfers. The data will load into the CAM during the idle time while packet data is being transferred according to a latched direction. In the case of system startup or node reassignment, the chip can enter a CAM load mode, where all packet transfers will be held, and data is loaded at 3 to 4 addresses per packet period.

Global Control Modes

There are four global control modes which determine the operating mode of the chip. These four modes are controlled by two pins on the chip. The first mode, is the Normal Operation mode. In this mode, the chip can perform packet transfers, as well as slow CAM loading, as described above.

Next there is the CAM Address Load mode, in which packet transfers are completely deactivated. This mode is meant uniquely for the fast CAM load, where data is loaded at 3 to 4 addresses per packet period.

A State Reset mode is provided, which deactivates the inputs to the chip and resets the FSMs. This mode is used in startup, or in debugging of the chip.

For testing purposes, a Test mode is used test the CAM and the FIFO contents. This is used in debugging of the chip.

3.5 Future Extensions

Our goal on this design was limited by the desire to stay within the allotted area for the assigned pad frame. This required several design choices which reduce the flexibility of the chip. As an expansion to this design, we would add a second pair of FIFOs for the inputs to reduce blocking situations. Also the output and address controls would be expanded to allow transmission through both outputs simultaneously.

More than the current eight CAM cells would be required to accommodate more than eight major nodes in the network, although some tricks can be played with the default direction stored in location 0. Only addresses going the other direction need to be explicitly stored in the CAM as long as the external controller can distinguish true address errors from the defaulted destinations. The four bit addresses used by the CAM can be considered only part of the total address, where the top four bits of the address are used for primary routing, and the remaining low order address bits are used for secondary routing once the packet reaches its destination area.

Chapter 4

VLSI Implementation Issues

As mentioned before, this project was originally designed in conjunction with the 6.371 Introduction to VLSI Systems course at MIT. The layout and simulation tools used were Mentor Graphics GDT and VHDL. The chip was designed using the MOSIS2p process, a 2 micron, p-well process, on a TinyChip pad frame. The approximate size of the padframe is 1800λ by 1800λ, where λ is 1 micron for a 2 micron process.

The design rules for this process were included into a technology file for the GDT tools, and were based on the MOSIS scalable CMOS design rules (rev 6).

4.1 Major Layout Blocks

4.1.1 CAM/RAM block

The CAM cell was based on a basic 6 transistor SRAM cell, with 3 added nFETs which act as a comparator and pull down for the match line. A schematic diagram of the CAM cell is in Figure 4-1. Sizing of the double inverters was made small so that the values could be easily overwritten. The match pull down transistor was made larger to ensure a quick pull down during a mismatch. The CAM/RAM block is implemented with a precharge/evaluate timing discipline. The match lines are precharged high during phi2, and then evaluated during the phi1 stage. In order to ensure that the match lines do not get accidently discharged, the bit and _bit

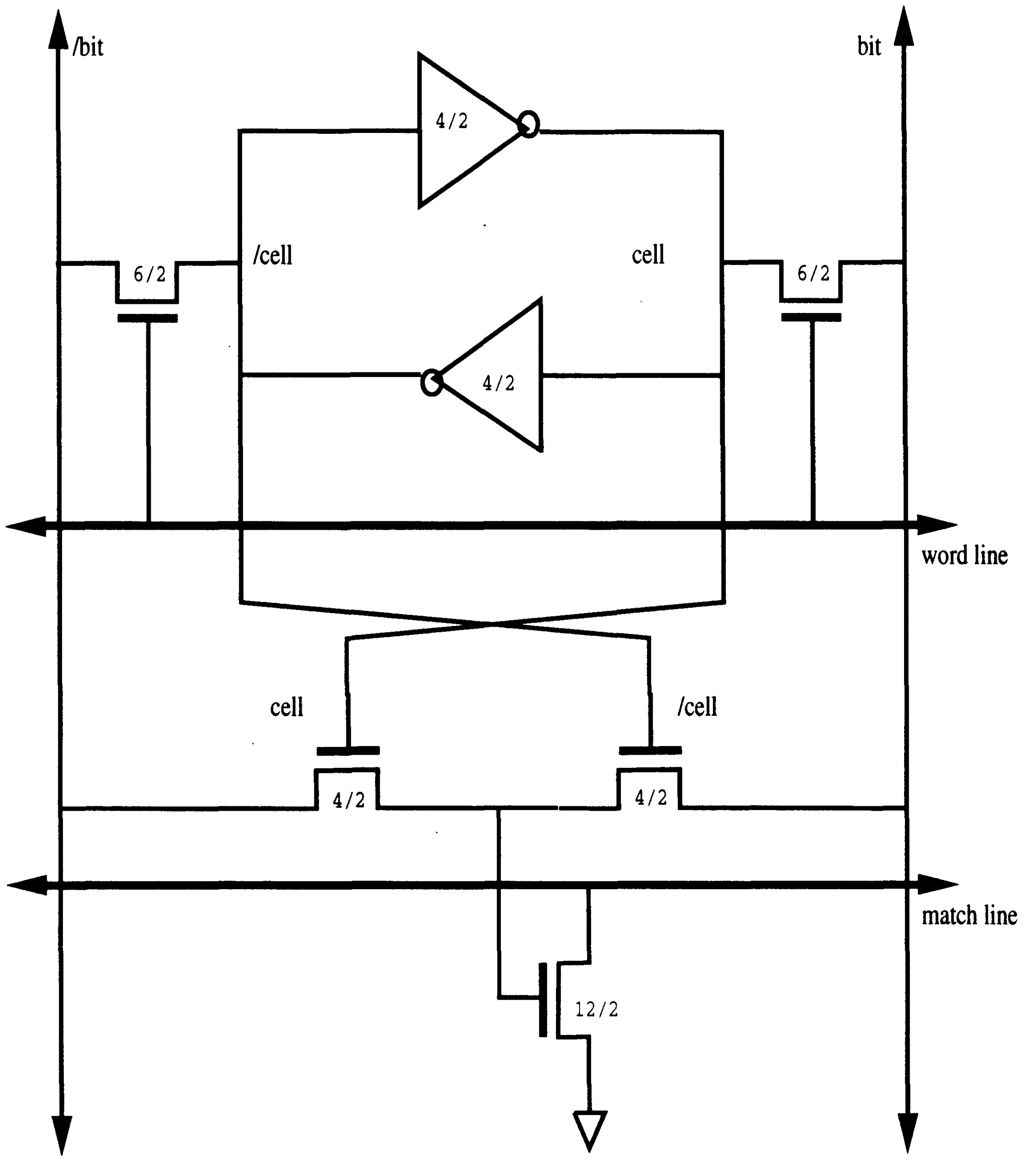

Figure 4-1: Schematic Transistor Level Diagram of CAM cell

lines in the CAM will be precharged low on phi2. In addition, the word line will be precharged low to ensure that the write line does not stay high into the next cycle. The bit and _bit lines therefore have to be qualified phi1 for proper operation of the circuit. Because the bit lines are received stable phi1 from the address mux, drivers were constructed to generate a _bit line for every bit of the address, as well as to ensure that the signals were driven qualified phi1. These were designed to abut with the top row of the CAM/RAM block.

The RAM was built using a standard 8 transistor SRAM cell, with pass transistors to break the feedback path. The RAM has a separate write bus from the load buffer and a read bus that feeds the direction latch. Sizing played a key role in the design of this block. The inverter which fed directly off of the write bus was made smaller, while the second inverter, which fed the read bus was made slightly stronger to drive the read pass gate and to properly restore the cell's value. In order to make this scheme work, two pass transistors were inserted in the paths from one transistor to another to alleviate fighting problems.

Design Review

During a write operation, the RAM is always written along with the CAM address information. In our original design, we had allotted for a mux in between the CAM and the RAM blocks to decode the 8 word lines and to reencode them for the RAM. However, after studying the situation further, this block was deemed unnecessary. In the interests of conserving space, the 8 word lines and the 8 match lines were run straight through an abutting cell to the RAM. One problem arose in the precharge value of the match lines. Because the match lines were being precharged high every cycle, connecting them directly to the RAM read lines would cause all RAM locations to be simultaneously read onto the bus during the precharge phase. This would cause major charge sharing on the read bus, and potentially leave the RAM locations vulnerable to a bit flip. Therefore, extra buffering was needed in between the CAM and the RAM to separate the CAM match lines from the RAM read lines. This buffering held the read lines low on phi2, and used a tristate buffer to enable the

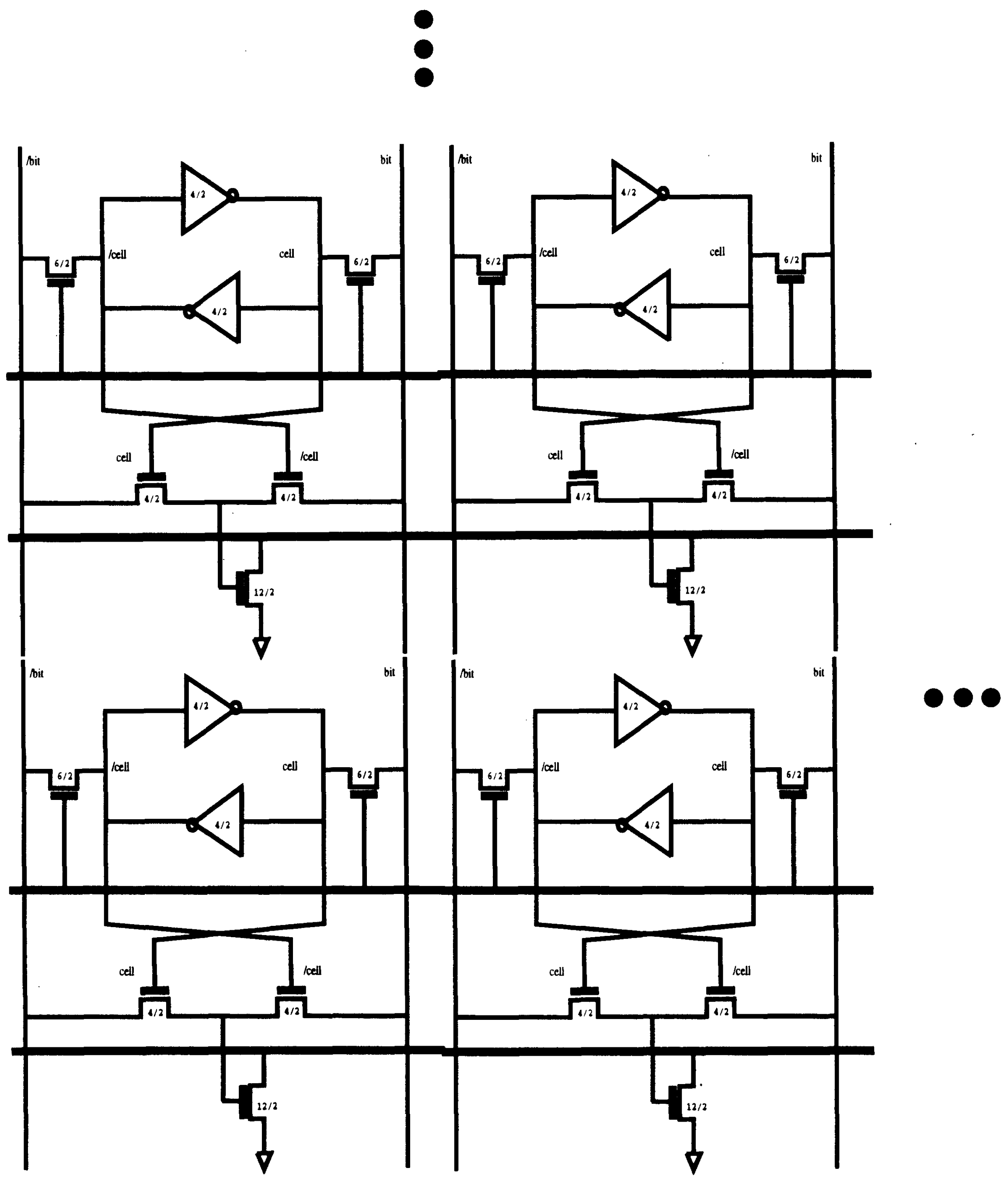

Figure 4-2: An Array of CAM cells

match lines through only after they had settled to their valid phi1 logic levels. In retrospect, this problem could probably have been avoided if the CAM had been designed in such a way that the match lines were precharged low and pulled high during a match. The way the CAM is implemented now also dissipates a large amount of switching power on every cycle. Of the eight CAM locations, at most one will produce a match on any given cycle. In the current design, seven of the lines will be charged high and discharged to a mismatch on every cycle, and the one match line will remain charged high. This causes unneeded switching current through the power and ground lines. Unfortunately this option presented itself too late in the design process to be properly implemented. It would have required a redesign of the entire CAM cell.

4.1.2 FIFO Buffers

The structure of the load buffer cell is identical to that of an input FIFO cell, with the exception of the parallel output terminals. The shift register that is used to load the buffer is abutted directly on to the buffer cells to conserve space and routing.

The FIFO consists of a basic 10 transistor 16x4 RAM with one write bus and one read bus. It was designed to have a 10ns access time for both read and write modes. The four column bits are loaded in parallel and cycled sequentially into and out of the 16 rows of memory The top 5 bits have an extra parallel output: bit 0 is interpreted as the Start of Packet signal, while the four address bits (bits 1-4) are stored in a register as an input for the CAM.

Row decoding for the read and write busses is controlled by a resettable shift register which cycles one enable bit to one row at a time. This shift register has been pitch matched to the standard fifo cell. An extra dummy register has been added which recycles the enable bit and outputs a FifoFull for the write decoder or End Of Data (EOD) for the read decoder. Each input FSM controls the (inc_fwr) signal for its fifo which increments/resets the write decoder The Read decoder is also a shift register controlled by the inc_frd signal which is selected by the address FSM and is triggered by the output FSM.

4.1.3 The Control FSMs

All of the control for the chip was represented by four distinct Finite State Machines (FSMs), which were implemented in the same manner. The control for the chip was broken up into four parts in order to simplify the design. Each separate FSM was therefore able to be implemented in four to five states. This was advantageous in the testing of the circuitry, as debugging could be attacked separately for each of the control blocks. In addition, the four control blocks were each compact in size, compared with one large FSM that could have been constructed. This provided us with flexibility in the placement of the blocks in the Floor Plan of the final chip. Each small control block was placed near the specific datapath blocks with which it interacted, and this strategy simplified the final routing of the chip a great deal.

The function of the four Control FSMs is discussed in Chapter 3. All of the FSMs were initially modeled in the VHDL Hardware description language at the behavioral level. This model was simulated to ensure the proper operation of the control system. After hand extraction of the truth tables and the gate level representation of the FSMs, a layout was generated from the gate level representation using a layout tool, AutoCells.

4.2 Implementation Issues

4.2.1 Floor Plan

The pinout and floor plan were coordinated to minimize the wiring of any four bit busses between inputs and the fifo, the fifo and load addresses to the address mux, the address mux to the CAM and the fifo busses to the output mux and the output pins. Each FSM was located near the area which it controls. The main long distance wiring is for the handshaking signals between controllers and the global clocks and mode signals as well as power and ground.

4.2.2 Power and Ground Distribution

The CAM block is the major power consuming portion of the project. As mentioned in the functional description of the CAM, the model used consumes an impractical amount of switching power. Through Lsim adept simulation, an estimate of the maximum switching current was found to be 0.6 mA in the match and ground lines. Because the current capacity of metal1 is 0.05 mA/um, this requires the Vdd, Ground and Match lines to have a wire width of 12 microns. In the present design, these lines are only 5um wide, but minor adjustments will be made to accommodate the 12 um lines during further testing for fabrication.

4.2.3 Clock Requirements and Distribution

A two phase, non-overlapping clock is used which will be supplied from an external source. This was done to allow debugging flexibility on the final chip. The maximum clock frequency is 20Mhz (50ns) All data transfers will be valid phase 1, for both transfers between internal blocks and also for external data. Phase 2 is used for precharging on the CAM/RAM cells and for latching in the registers and FSMs.

4.3 Testing

4.3.1 Control Blocks

All FSMs were tested in the same manner. First a behavioral model was written and fully simulated in VHDL. Upon completion of the VHDL simulation, the state equations were hand extracted by drawing state diagrams and state transition tables. Gate level schematics were then designed using the standard cell library set up in Led. The synthesis tool AutoCells was then used to convert the gate level schematic to a standard cell layout.Once the FSM standard cell layouts were generated, each was tested in Lsim adept mode and compared to the VHDL simulation results.

4.3.2 Datapath Blocks

The functional blocks that were simulated in Lsim are: the CAM/RAM block, the input FIFOs, the output mux and all of the smaller mux and driver cells. Because the CAM cell was the most uncertain in design, it was constructed in schematic mode in Led and fully simulated in Lsim adept mode. Representative capacitances were added to all of the busses to ensure that the cell would function in the full array. In addition, a single word cell CAM block was first simulated and tested in schematic mode Led and adept mode Lsim. The rest of the blocks were entered directly into layout mode in Led and fully simulated in Lsim adept mode. Each of the blocks was thoroughly tested by itself before placing it into the pad frame.

4.3.3 Verification of Fabricated Chip

During Test mode, the fifo address requests are disabled so that addresses can be entered from the load buffer using the normal load input protocol. These addresses are then evaluated by the CAM and the location of the CAM cell is output sequentially on the cam_loc pin. The direction bit is output on the external DIR pin. The external controller must cycle through the addresses to check the match conditions. It is responsible for preventing the same address from being loaded into more than one location. For networks with less than eight nodes, one address should be reserved as a default address which can be used for any unused CAM cell locations. Fifo testing also uses the Test mode, which will force a straight data pass through West to East and South to North during the testing. This removes any dependence on the CAM and simplifies testing of the data path.

Chapter 5

Conclusion

The implementation issues which I researched over the course of this thesis cover several different areas in Network Routing. In order to gain as broad of an understanding of this topic as possible, I studied issues ranging from transmission and timing effects for an input port, to multiprocessor network theory, to VLSI chip design of a functional router.

My work included secondary research into articles and papers on relevant topics, as well as a large amount of design work in the VLSI implementation of the CAM based router. Because of this, I was able to apply existing research to an idea of my own, and to quickly gain a current and relevant background in the field of network routing. I feel that this thesis succeeded in its goal of presenting many of the issues surrounding network design for multiprocessing.

As it stands, my design is a very useful starting point for a router design. It has all of the basic components of a functional router, and addresses several of the network issues that are needed to build a modern router for multiprocessing. In addition, the CAD design work for the VLSI implementation is a solid base upon which to further the design.

Unfortunately, the fabrication of the chip that is described in this thesis is not complete. Due to constraints on time and resources, I was unable to get the chip fully tested and ready for fabrication. However, fabrication was never a goal for this project because of the large time delay in getting fabricated chips back from the Fab.

Because of the limited time of my degree, I would not have been able to test the fabricated chips while I was still at M.I.T.

If future work on the project is considered, it would definitely include the redesign of the CAM cells as described in Chapter 4, as well as a more in depth look into the routing issues covered in Chapter 2. I feel that this initial design could be greatly improved upon if more network issues were accounted for in the hardware design.

Even without these changes, the CAM based router project was a success in meeting its design goal to include all of the basic elements of a functional router. At the hardware simulation level, the chip is fully functional as specified and is a good start at the design of a network router.

Appendix A

Timing Issues for ARCTIC

This appendix was added to highlight my previous work with implementation issues with ARCTIC [5]. The timing issues that I studied on the ARCTIC project are both relevant and significant to the design of any packet router. Although these issues were not specifically addressed in the design of the CAM based router, the research that I did played an underlying role in how that router was implemented. The following chapter is excerpted from my Bachelor's thesis [2].

The hardware blocks that I dealt with are the output unit, the input unit, and the chip-to-chip interconnection. We are concerned with the transmission of signals between the output unit of one chip and the input unit of another. The communication signals between these blocks are two clock lines, sixteen data lines and one frame bit. In this discussion, we are concerned with the timing issues of properly latching the data for the rest of the chip. A block diagram of this is shown in Figure A-1.

The data must pass from one chip, through the output unit, through a transmission line signified by the pc board trace, and to the input unit of the next chip. Once is it received at the input unit, the data is latched by a register in order to be used by the digital circuitry of the chip.

The problem lies in the fact that the clock signal and the data lines are both transmitted together along this path, and due to skew and other factors, they may not arrive in the same order as they left. In the case where the clock signal gets delayed beyond the data line, the data will be incorrectly latched into the register.

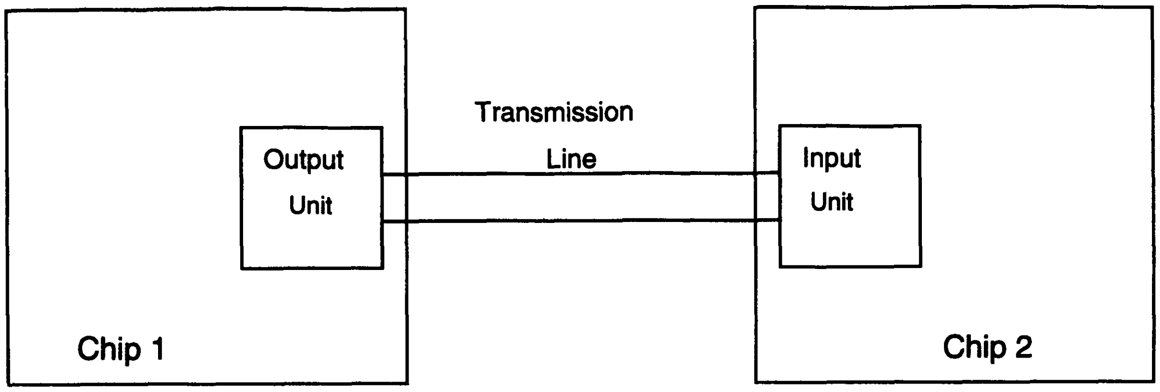

Figure A-1: Block Diagram of the System

For this reason, we must pay careful attention to the timing of the clock in relation to the data lines.

A.1 The Clock

Two 50 Mhz clock lines are used which are 180 degrees out of phase with each other. This configuration will be used later in the design process to demultiplex the data lines in the input section from 16 to 32 lines. Having two phases of the clock allows us to latch 16 bits of data on the falling edge of the first phase, and then to latch the next 16 bits of data on the falling edge of the second phase. This results in latching of data every half-clock period or 10 ns. Figure A-2 illustrates the clock period for both phases of the clock.

The flip-flops are triggered on the falling edge of the clock, so we must concern ourselves with the window of time between the falling edge of one clock and the falling edge of the next. The specifications of the clock period are such that the duration time from the falling edge of the positive clock to the falling edge of the negative clock is 10 ns. plus or minus 1.5 ns. In addition, the rising edge of each phase should occur 10 ns. plus or minus 3 ns. after the falling edge of that phase. This ensures that the total cycle time is very close to 20 ns.

As a result of this uneven clock period, the rising edge of one clock does not

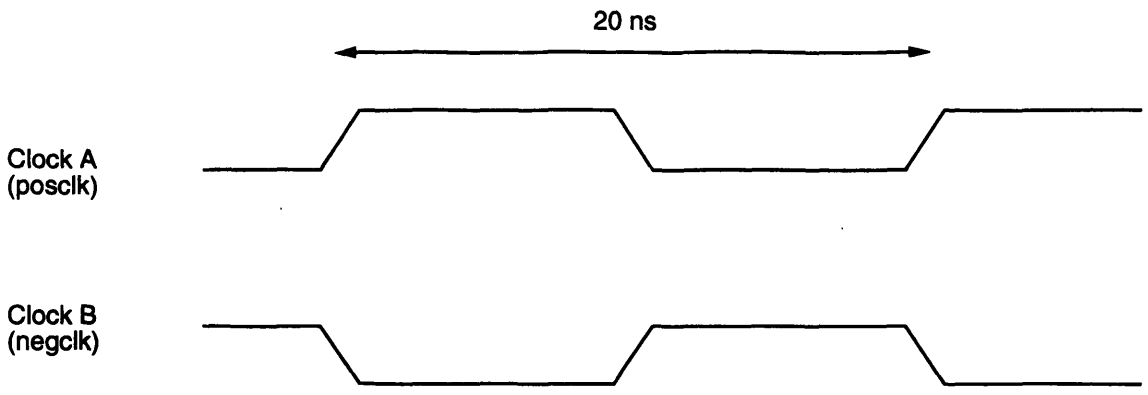

Figure A-2: Two Phases of the Clock

always occur at the falling edge of the other, even though they are supposed to be 180 degrees out of phase. In addition, the time from one falling clock edge to the next is not always exactly equal to 10 ns. In this situation, the minimum time window must be used to determine the timing issues regarding the transmission, reception and latching of one bit of data. In our case, this minimum time window is 8500 ps.

A timing diagram of the whole system can be seen in Figure A-3.

A.2 The Output Unit

The Output unit is a rather simple block, consisting of the output pad, a cell, and some core circuitry. In the initial design of the system, it was decided that the clock and the data would transition simultaneously at the transmit pad. For this reason, the functionality of the output unit is relatively simple.

The output unit will begin to assert data a maximum of 1500 ps. after the previous falling edge of the clock. This results in a minimum time of 7000 ps. before the clock edge on which it will be latched. In addition, this data must remain stable at the transmit pads for at least 250 ps. after the falling edge of the clock. The data is therefore stable at the output pads for a total time window of 7250 ps. in the worst case. That is, 7000 ps. before the given clock edge, and 250 ps. after. These timing diagrams are clearly illustrated in Figure A-3.

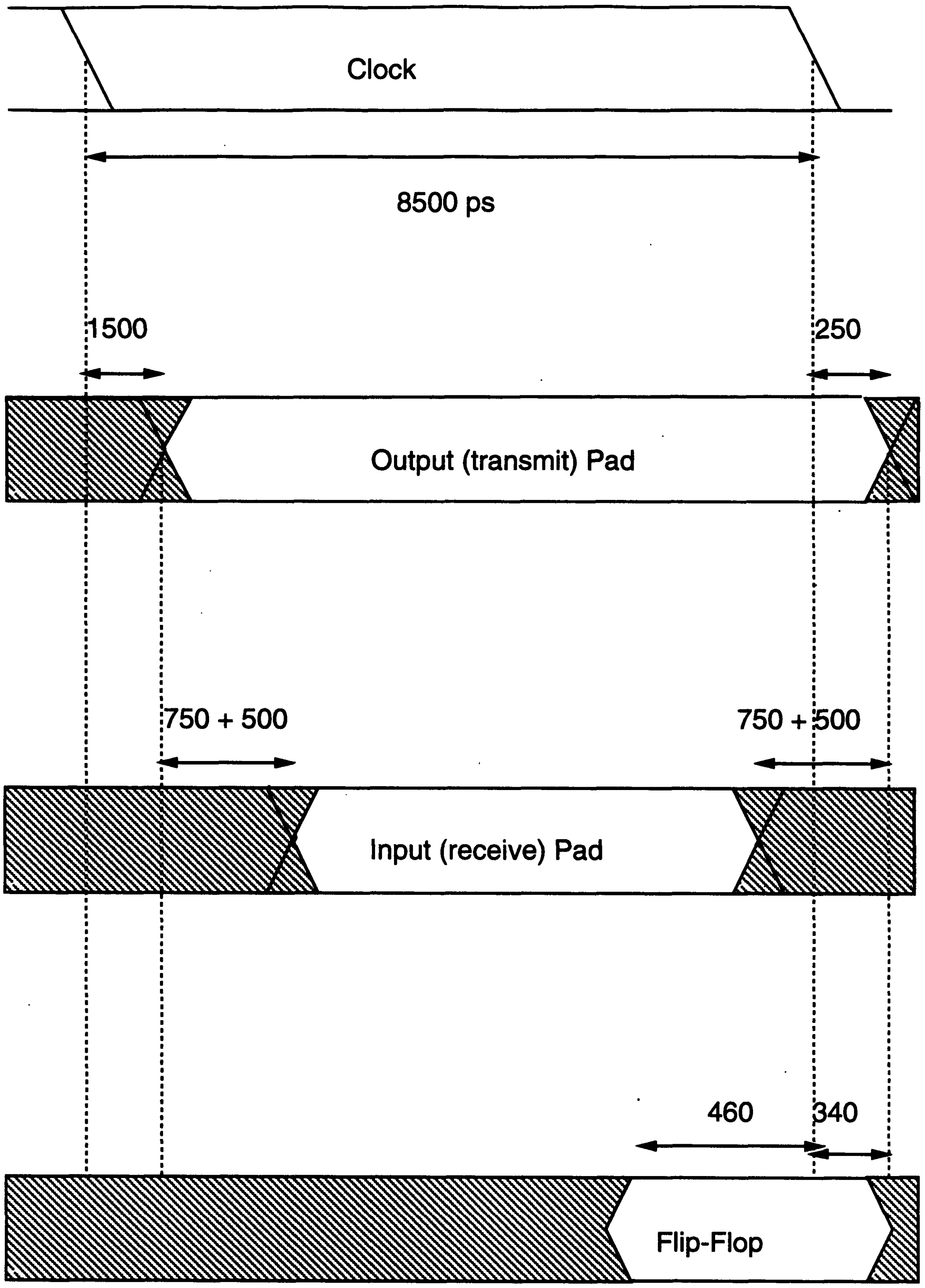

Figure A-3: Timing Scheme for Output pad, Input Pad, and Input Flip-Flop

A.3 The Input Unit

The input unit is a bit more complicated, as it consists of the input pad, a cell, some core circuitry, and a register to latch the data. The purpose of the input unit is to receive the data from the output unit, and latch it so that it may be used by the rest of the chip. To do this, it will implement a delay line and clocking strategy to translate the 16 data lines from the transmission line to 32 lines for the rest of the chip. See Figure A-4 for a block diagram of the input section.

A.3.1 The Input Pad

The input pad handles the data at the other end of the transmission line. Refer to Figure A-1 for a block diagram of the system. Because of delays in the transmission of signals from the output unit, the data and the clock may be received in a different order than they were sent. If the data is delayed relative to the clock, then proper latching will not occur at the input flip-flops. We must therefore allow for estimated time variances in determining the timing scheme for the system. There are two ways in which the data may be delayed relative to the clock at the input pad.

The first way in which data could be delayed relative to the clock is due to the physical differences in the length and electrical characteristics in the traces. Because the length of the clock trace may be slightly smaller than the data trace on the PC board, the data may be delayed by an arbitrary amount. In addition, slight differences in the electrical characteristics of the clock trace and the data trace could cause a similar delay. We have made a conservative estimate for this case, and allowed for 500 ps. of skew between the two signals. This allowance can be seen in the timing diagram in Figure A-3.

The second factor which can cause a potential time delay between the clock and the data signals at the receive pad is noise. Because of noise which may enter the system while the data is switching from high to low or from low to high, the time may vary at which the data crosses the valid logic level. We can estimate a value for the time difference from the signal rise time, which may be about 750 mV in

3 nanoseconds, or 250 mV/ns. If we allow for the voltage to fluctuate by plus or minus 190 mV at the receive pads, then the corresponding time factor can be found by multiplying the switching rate by the voltage fluctuation. In the worst case, this would give us approximately 750 ps. This 750 ps. is added to the 500 ps. from above, and factored into the timing scheme.

In the timing diagram, the data is therefore taken to be valid at the receive pad a maximum of 1250 ps. after it attains validity at the transmit pad. It will remain valid until 1250 ps. before it goes invalid at the transmit pad. In this way, the timing difference is accounted for at both ends of the valid time window at the receive pad.

As can be seen from the timing diagram, the data is then valid at the input pad for a total time window of 4750 ps. This window begins 5750 ps. before the falling clock edge on which the data is latched and ends 1000 ps. before the same clock edge.

A.3.2 The Input Register

The input register is used to latch the data from the input pad, so that it may be synchronous with the digital circuitry within the chip. We have chosen to implement the input register using flip-flops of the type Motorola H4CDFF1A, which are scan D flip-flops. In order to satisfy the setup time of the flip-flop, the incoming data must be valid at the D pin for a minimum of 460 ps. before the falling edge of the clock. In addition, the hold time of the flip-flop requires that the data remain valid at the input pad for a minimum of 340 ns.

The minimum time window for which data must be valid at the flip-flop therefore extends from 460 ns. before the falling clock edge to 340 ns. after it, or a total window of 800 ns. An example of this can be seen in Figure A-3.

A.3.3 The Delay Line

As we can clearly see from the timing diagram, we must implement the delay line shown in Figure A-4 in order to properly latch the data for the system. Choosing the value for this delay line is non-trivial due to variations in process, temperature and

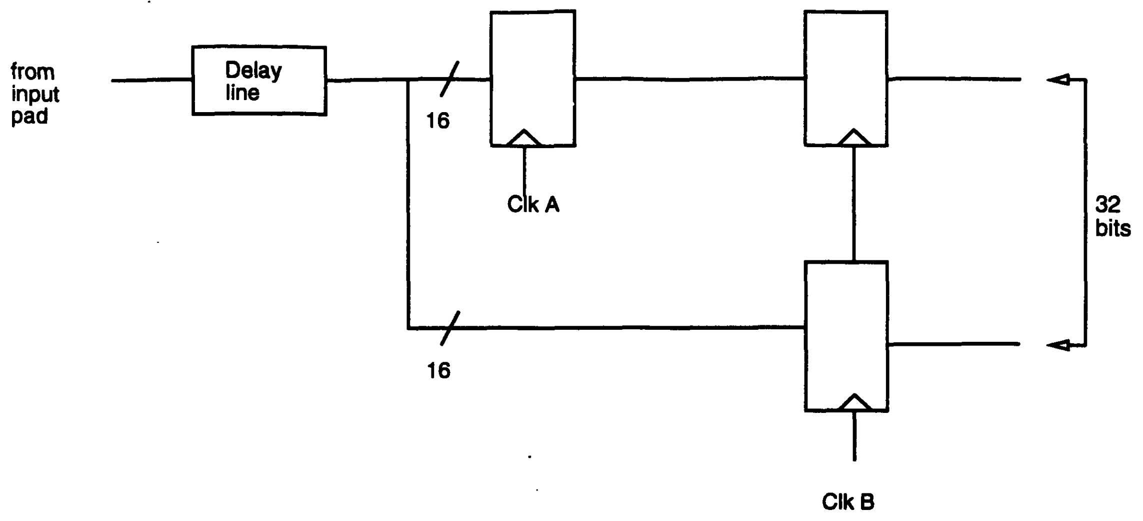

Figure A-4: Block Diagram of the Input Section

voltage. These variations result in quite different values for the delay, depending on the specific process, temperature and voltage in each case.

Once the actual delay range has been determined, we must consult the Process, Temperature, Voltage Multipliers chart, which can be found in [7, p.3-8]. This chart gives multipliers to determine the nominal range of values needed for the delay, depending on the best and worst case voltage and temperature for the system.

From the timing diagrams in Figure A-3, we can easily determine the limits for the value of the delay line. If we look at the valid window at the input pad, we see that it begins 5750 ps. before the falling edge of the clock, and ends 1000 ps. before it. Therefore, by delaying the data at the input pad by at least 1340 ps., we can satisfy the hold time of the DFF1A Flip-Flop.

Similarly, the maximum delay can be found by subtracting the setup time, or 460 ps. from the beginning of the valid time window, or 5750 ps. before the falling edge of the clock. The data should therefore be delayed no more than 5290 ps. in order to satisfy the setup time for proper latching to occur. The real value for the delay line is thus between 1340 and 5290 ps. This can be verified by studying the timing diagram in Figure A-3.

Next, the values of voltage and temperature in the extreme case are needed to determine the nominal delay line value. The voltage and temperature has been determined to vary from a best case of 0 degrees(C) and 5.5V to a worst case of 110 degrees(C) and 4.5V. These values correspond to a best case multiplier of 0.67 and a worst case multiplier of 1.74.

In order to find an appropriate nominal value for the delay line, we must calculate a value such that 0.67 times the delay value will still satisfy the hold time, and 1.74 times the value will still satisfy the setup time. By simple calculation, we can see that the nominal value for the delay line should be no smaller than 2000 ps., and no larger than 3040 ps.

Appendix B

Transmission Issues for ARCTIC

Again, this chapter outlines some of the work that I did toward my S.B. thesis on the input section of the ARCTIC router [5]. Like Appendix A, I felt that the reaults of this research were relevant to this discussion of implementation issues for packet routing in multiprocessor networks.

In this section, I will be concerned with the transmission line which connects the output unit of one ARCTIC chip to the input unit of another. Refer to Figure A-1 for a block diagram of the system. The transmission line environment consisted of a GTL driver, the printed circuit (PC) board trace, and the multi-chip module trace. While the transmission line in the block diagram represents all of these entities, when examined more closely, they are very distinct units. For the purposes of this project, the GTL driver was treated as a black box with a given set of characteristics. The electrical characteristics of the traces, however, were examined in more detail. All three components of the transmission line block were modeled using hand calculations and given specifications, as well as HSPICE software models.

B.1 Initial Characterization of Trace

The PC board trace consisted of a copper trace engineered to a 50 ohm transmission line environment. The MCM trace used a molybdenum trace to obtain a similar 50 ohm transmission line environment. Both the PC board and MCM trace were

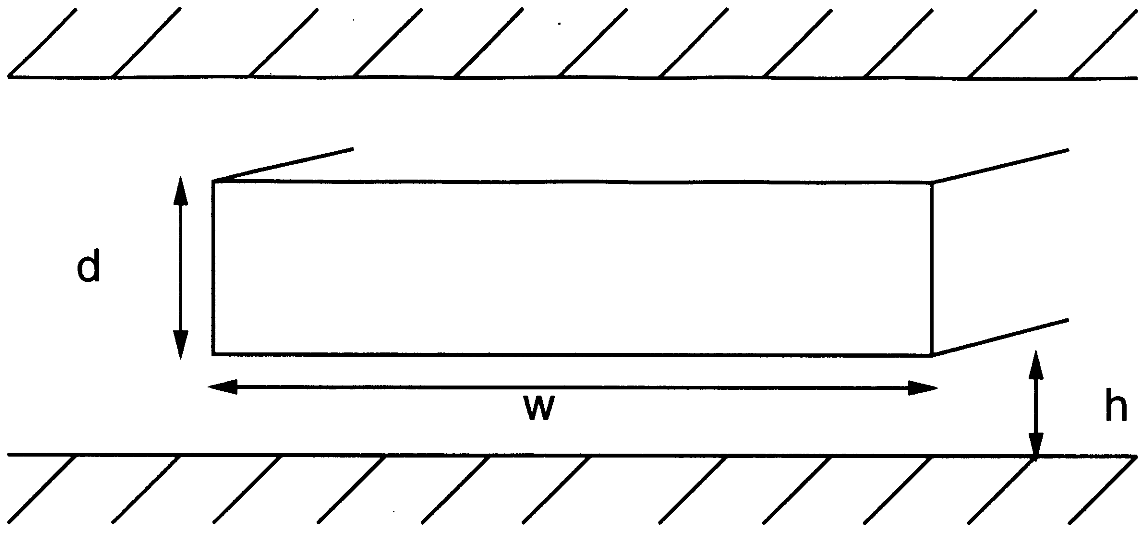

Figure B-1: Stripline construction of the PC Board and MCM Trace

modeled using a tri-plate stripline construction, which can be seen in Figure B-1. There are two ground planes which surround a center conductor, whose dimensions are given by the variables d, w, and h.

The factors which contribute to the transmission characteristics of the trace are the resistivity of the center conductor, the dielectric constant of the material surrounding the conductor, the transmission frequency, and the dimensions of the conductor. As many of the former factors are accounted for in the specifications of this project, the only variable factor was the dimensions of the conductor. Using an iterative design approach, I first modeled the system on HSPICE, and then varied the conductor dimensions to observe the results. I will therefore begin discussion using the model where h is four mils, w is 12 mils, and d is four mils, where the dimesions are as given in Figure B-1.

B.2 Calculation of Termination

The first task of this section of the project was to determine the conductor height that would most closely correspond to an impedance of 50 ohms. As the termination resistance was chosen to be 50 ohms, we needed the line impedance to approximate this value as closely as possible in order to eliminate reflections of the signal.

The equation for the impedance of a stripline was found to be

$$Z_0 = \frac{h}{8}\frac{377}{2X} \tag{B.1}$$

where X is defined as

$$1 + (\frac{2h}{8\pi})(1 + \ln(\frac{8\pi}{h})) \tag{B.2}$$

With a value for h of 4 mils, the characteristic impedance comes out to be very close to 50 ohms. This approximation was used in developing the HPICE model for the trace. See [6] for more background regarding this equation.

B.3 Skin Effect Considerations

Due to the extremely high frequencies in this application, we must take into consideration the skin depth of the conductor. Because the conductivity of the copper is finite, at high frequencies the electromagnetic fields can only penetrate a small distance into the copper. The distance which the fields can penetrate at a given frequency is referred to as the skin depth. The skin effect, as it is called, has a significant impact on the resistance of the trace.

Using [9],we can calculate the skin depth (δ) for the copper trace by using the equation

$$\delta^2 = \frac{2}{\omega\mu\sigma} \tag{B.3}$$

where

$$\mu \approx \mu_0 = 4\pi \times 10^{-7} \tag{B.4}$$

and $\sigma = 5.8 \times 10^7 \Omega^{-1} \cdot m^{-1}$ for copper. For a frequency of 400MHz, we get $\omega = 2\pi f = 800\pi \times 10^6$ rad/sec. Plugging these values into the equation for δ , and taking the square root, we get $\delta = 0.13$ mil. This is the skin depth for copper at 400 MHz.

Continuing on, we would like to then find the resistance of the copper trace. We use the equation

$$R = \frac{1}{A\sigma}\Omega in^{-1} \tag{B.5}$$

from [6]. We can then use this value for R to get α, or the attenuation of the line.

$$\alpha = \frac{R}{2Z_0} + \pi f C Z_0 \tan\delta \tag{B.6}$$

where in this case, $\tan\delta = 0.03$ for the dielectric used in this trace. Calculating this by substituting the known values into the equation, we see that $\alpha = 1.12 dB/ft.$ for 400MHz.

B.4 Analysis at Harmonic Frequencies

Although the system clock for this project runs at 50 MHz, it was necessary to test the transmission of the line at several harmonic frequencies as well. We were mainly concerned with signals at 100, 200 and 400 MHz. Using the method that is explained in Section B.3, we can easily calculate the values of α for the 200MHz and 100MHz cases. By plugging into the formulas given in Section B.3, we see that

$$\alpha_{200MHz} = 0.653 dB/ft \tag{B.7}$$

and

$$\alpha_{100MHz} = 0.393 dB/ft. \tag{B.8}$$

Finding these values for α is useful in setting up the HSPICE model for the system. The information derived from this model will be used to help understand the attenuation of the system as a function of frequency. This attenuation in the high frequency components of the signal have a significant impact on the quality of the edges for the the 50MHz digital data.

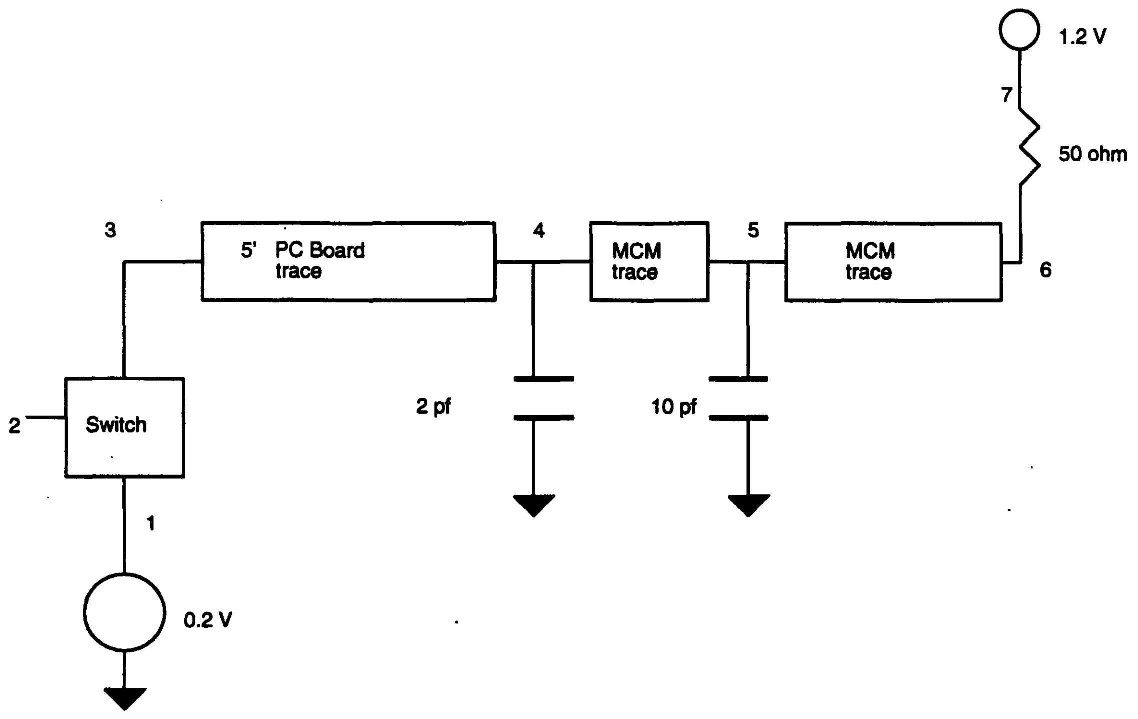

Figure B-2: Diagram of transmission line system

B.5 Modeling the Entire Transmission Line

To model the environment of chip-to-chip interconnection, we need the copper PC Board trace, the Molybdenum MCM trace, the switch driver, and various capacitances of the system. To model the MCM trace, I went through the same steps as in the previous sections, substituting in the conductance of Molybdenum (σ_{Mo}) for the conductance of Copper(σ_{Cu}).

A diagram of the final HSPICE model of the transmission line system is shown in Figure B-2. The entire system is tied between 0.2V and 1.2V to represent the low voltage swing of the system. A GTL driver was represented by the switch, which was then modeled on HSPICE.

Overall, this model was used to give some useful data about the system. However, the design is expected to keep evolving, and changes will be made to the original model.

Bibliography

[1] Christopher Frank Joerg. *Design and Implementation of a Packet Switched Routing Chip.* Technical Report MIT/LCS/TR-482, MIT Laboratory for Computer Science, December 1990.

[2] Gowri Rao. *Transmission and Timing Issues for ARCTIC transmit and receive Apparatus* S.B. thesis, MIT May 1993.

[3] Gowri Rao and Roberta Bailey. *A Packet Router Chip with Content Addressable Memory* 6.371 Design Project December 8. 1993

[4] Andre DeHon. *Robust High-Speed Network Design for Large Scale Multiprocessing* Technical Report MIT/AI/TR-1445, MIT AI Lab, September 1993.

[5] G. A. Boughton, et al. *ARCTIC User's Manual* March 9, 1993. Unpublished.

[6] Terman. *Electronic and Radio Engineering* McGraw Hill.

[7] Tummala. *Microelectronics Packaging Handbook* Van Nostrand Reinhold.

[8] Ruth Massey. *Motorola H4C Series Design Reference Guide.* 1992. Motorola Inc.

[9] H. A. Haus. *Electromagnetic Fields and Energy* MIT Press.

[10] H.B. Bakoglu. *Circuits, Interconnects and Packaging for VLSI* Addison-Wesley, 1990.

[11] David C. Wyland. *VLSI CAM applications - An Overview* Wescon/89 Conference Record, 1989.

[12] Z. Amitai. *Address Filtering in FDDI LAN bridges (the CAM solution* Wescon/89 Conference Record, 1989.

[13] T. Pei and C. Zukowski. *VLSI Implementations of routing tables: tries and CAMs* IEEE Infocom '91 Proceedings, vol.2, 1991.

[14] Weste and Eshraghian. *Principles of CMOS VLSI Design* Addison-Wesley, 1992.

[15] Fried, J. *A VLSI chip set for burst and fast ATM switching* IEEE International Conference on Communications, vol.1, 1989.

[16] R.S. Nikhil, G.M. Papadopoulos and Arvind. **T: A Multithreaded Massively Parallel Architecture* Proceedings of the 19th International Symposium on Computer Architecture, May 1992.