INFORMATION TO USERS

While the most advanced technology has been used to photograph and reproduce this manuscript, the quality of the reproduction is heavily dependent upon the quality of the material submitted. For example:

- Manuscript pages may have indistinct print. In such cases, the best available copy has been filmed.

- Manuscripts may not always be complete. In such cases, a note will indicate that it is not possible to obtain missing pages.

- Copyrighted material may have been removed from the manuscript. In such cases, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, and charts) are photographed by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each oversize page is also filmed as one exposure and is available, for an additional charge, as a standard 35mm slide or as a 17” x 23” black and white photographic print.

Most photographs reproduce acceptably on positive microfilm or microfiche but lack the clarity on xerographic copies made from the microfilm. For an additional charge, 35mm slides of 6” x 9” black and white photographic prints are available for any photographs or illustrations that cannot be reproduced satisfactorily by xerography.
Reduced bandwidth delta networks

Harper, David Tennyson, III, Ph.D.
Rice University, 1987
PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark.

1. Glossy photographs or pages
2. Colored illustrations, paper or print
3. Photographs with dark background
4. Illustrations are poor copy
5. Pages with black marks, not original copy
6. Print shows through as there is text on both sides of page
7. Indistinct, broken or small print on several pages
8. Print exceeds margin requirements
9. Tightly bound copy with print lost in spine
10. Computer printout pages with indistinct print
11. Page(s) lacking when material received, and not available from school or author.
12. Page(s) seem to be missing in numbering only as text follows.
13. Two pages numbered. Text follows.
14. Curling and wrinkled pages
15. Dissertation contains pages with print at a slant, filmed as received
16. Other

University
Microfilms
International
RICE UNIVERSITY

REDUCED BANDWIDTH DELTA NETWORKS

by

DAVID TENNYSON HARPER III

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

DOCTOR OF PHILOSOPHY

APPROVED, THESIS COMMITTEE:

J. Robert Jump, Professor of
Electrical and Computer Engineering,
Director

J. B. Sinclair
J. B. Sinclair, Associate Professor of
Electrical and Computer Engineering

W. Zwaanepoel
W. Zwaanepoel, Assistant Professor of
Computer Science

Houston, Texas
May, 1987
Abstract

As multiprocessor architectures become more prevalent, understanding the performance of interconnection networks becomes more critical. Because the cost of these networks is high, it is desirable to use the least sophisticated network which will not degrade system performance.

By considering a more general model of delta networks it is shown that the full bandwidth of these networks cannot be utilized in many situations and that the amount of network bandwidth that can be utilized varies from application to application. Because the bandwidth required to support a given application does vary, it is desirable to evaluate the performance of a family of networks whose members have lower bandwidths than do delta networks. From this family of hybrid networks the choice of interconnection network can be made such that system cost is minimized while system performance remains unchanged.

Since VLSI is the predominant implementation technique in current computer systems the constraints imposed on network design by VLSI must also be considered. In particular, the I/O bottleneck caused by pin limitations may force the network to be partitioned over multiple VLSI packages. How the network is partitioned greatly impacts the level of system performance. Additionally, implementation of networks in VLSI presents an alternative method of reducing system cost. This is accomplished through reduction in network data path width. Under some conditions this technique is a better method of reducing system cost than is the use of hybrid networks.
Acknowledgements

I would like to express my sincere thanks to my research committee, Dr. J. R. Jump, Dr. J. B. Sinclair, and Dr. W. Zwaenepoel. Their comments, suggestions, and corrections have been invaluable. I am particularly in debt to Dr. J. R. Jump who has given me such superb guidance over the past five years. I could not have received better direction for my investigations.

Thanks to all of my friends who have shared my joys and frustrations. I hope Kshitij, Rick, Sridhar, Hemant, and Allan find equally sympathetic advisors. Thanks for the technical discussions and for the beers at Vahalla; both were indispensable. And many thanks to Dr. Bill Wilson for time and time again keeping me on a level keel. The peace and tranquility of Boquillas Canyon will always be in my mind.

To my parents I can not give enough credit. Without their examples of achievement I question whether I would have persevered in pursuit of this degree. Without their foresight in laying the foundations of my education I would not have the breadth of experience that I have needed. Any successes I have in the future will be simple reflections of their efforts of the past.

Finalemente, a mi Barbara. Thank you for the constant encouragement, understanding, and love you’ve given me. The colors in my eyes are more brilliant for knowing you. Te quiero siempre.
# Table of Contents

## CHAPTER 1  Introduction
- Parallel Computers .............................................. 1
- Multiprocessors .................................................. 3
- Interconnection Networks ....................................... 4
- Problem Statement ............................................... 9
- Dissertation Outline .......................................... 10

## CHAPTER 2  Motivation
- System Model ................................................... 11
- System Bottlenecks ............................................ 13
- Performance Measures ......................................... 13
- Large Reply Times .............................................. 14
- Insufficient Concurrency ....................................... 15
- Implementation Problems ..................................... 16
- Summary and Proposals ....................................... 17

## CHAPTER 3  Address Routed Networks
- Models ............................................................... 18
- Switch Model ..................................................... 19
- Resource Model .................................................. 24
- Network Model ................................................... 26
- Model Verification .............................................. 29
- Model Inaccuracies ............................................ 30
- Hybrid Networks ................................................. 32
<table>
<thead>
<tr>
<th>Chapter/Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid Network Performance</td>
<td>35</td>
</tr>
<tr>
<td>Delay Driven Simulation</td>
<td>41</td>
</tr>
<tr>
<td>Architectural Evaluation Using a Delay Driven Load Model</td>
<td>44</td>
</tr>
<tr>
<td>Analysis: $O_{max} = 1$</td>
<td>45</td>
</tr>
<tr>
<td>Analysis: $O_{max} &gt; 1$</td>
<td>47</td>
</tr>
<tr>
<td>Summary</td>
<td>49</td>
</tr>
<tr>
<td><strong>CHAPTER 4 Resource Scheduling Interconnection Networks</strong></td>
<td>52</td>
</tr>
<tr>
<td>Related Work</td>
<td>53</td>
</tr>
<tr>
<td>Multistage RSIN Model</td>
<td>54</td>
</tr>
<tr>
<td>Hybrid Network RSIN Performance</td>
<td>59</td>
</tr>
<tr>
<td>Delay Driven RSIN Performance</td>
<td>60</td>
</tr>
<tr>
<td>Summary</td>
<td>62</td>
</tr>
<tr>
<td><strong>CHAPTER 5 Network Implementations under VLSI Constraints</strong></td>
<td>64</td>
</tr>
<tr>
<td>Previous Work and Chapter Overview</td>
<td>64</td>
</tr>
<tr>
<td>Narrow Delta Networks - Wide Buses</td>
<td>67</td>
</tr>
<tr>
<td>Network Partitioning</td>
<td>69</td>
</tr>
<tr>
<td>Narrow Path Bandwidth Reduction</td>
<td>78</td>
</tr>
<tr>
<td>Hybrid and Narrow Network Comparison</td>
<td>81</td>
</tr>
<tr>
<td>Summary</td>
<td>84</td>
</tr>
<tr>
<td><strong>CHAPTER 6 Conclusions</strong></td>
<td>86</td>
</tr>
<tr>
<td>Extensions and Elaborations</td>
<td>89</td>
</tr>
</tbody>
</table>
CHAPTER 1

Introduction

Examples of parallelism are seen everywhere in daily living. There are multiple copies of the same newspaper. The assembly line in a factory operates in a parallel fashion. The numerous routes that can be traveled to move from one area of a city to another area are yet another example. The reasons for these uses of parallelism are obvious: the assembly line improves productivity over that of individual workers because each worker need not know the complexities of others' tasks; multiple copies of the same newspaper avoids fistfights at the coffee machine in the morning; parallel routes between two locations allow continued travel even if one path is closed due to an accident or repair work. It is quite impossible to imagine society functioning in a serial manner.

Today's computational problems in knowledge engineering, meteorology, biomedical imaging, and other disciplines are demanding increased performance from computer systems [1]. These demands for increases in performance have outstripped even the exponential increase in component performance that has occurred in the last decade and the proximity of fundamental physical limits on device performance to current technologies leaves increased parallelism as a primary option for performance improvement in new architectures.

Parallel Computers

The examples listed above demonstrate three reasons for using parallelism that are also applicable to the design of parallel computer architectures: concurrent solution of multiple problems, improved system throughput, and fault-tolerance. These are not new ideas. The Tandem Corporation for many years has used duplicate hardware modules to improve system reliability [2] NASA space missions use independently operating computers to solve a particular problem and then to "vote" on the correct answer. Pipelines, used to improve system throughput, have become almost mandatory in today's high-performance systems even at the microprocessor level [3,4]. A more current example of
SIMD architecture is the highly complex Connection Machine [5]. The SIMD class of architectures is characterized by multiple processing elements, or PEs, each performing identical operations on different data. The PEs are usually organized as a one or two dimensional array. Sequencing of operations is done by a single control unit. The BSP has a linear array of 16 PEs. The Connection Machine has thousands of PEs connected in a complex interconnection pattern. A diagram of an SIMD machine organization is shown in Figure 1.1.

MIMD architectures are more recent and have appeared on the commercial market only in the past several years [6, 7]. In general, MIMD machines not only operate on different sets of data but perform different operations as well. The computational units are usually more complex than the

![Diagram](image-url)  

Figure 1.1: Block Diagram of an SIMD Architecture
SIMD's PEs. This is expected. Since there is no single control unit to sequence operations the sequencing is done at each processor.

Multiprocessors

In this dissertation a multiprocessor refers to a computer architecture which is considered to be in the MIMD class. Multiprocessors can be additionally classified by the degree of interaction between processors in the system. This is known as the coupling of the system. A multiprocessor is classified as loosely-coupled if the processors do not interact very much and as tightly-coupled if the interaction is relatively strong. The boundary between these two classes is somewhat blurred [8].

Processors can interact with each other both explicitly and implicitly. An example of explicit interaction is the need for communication between processes executing on different processors. A parallel algorithm may require one processor to wait for information being computed on another processor. Explicit interaction is generally determined by the use of the machine and therefore is a dynamic quantity. Implicit interaction on the other hand, occurs in the system when multiple processors require a common resource. This type of interaction results from the architecture of the system and is generally fixed at design time. It is the implicit interaction which determines the architecture's classification. Example architectures are shown in Figure 1.2. The degree to which resources must be

---

Figure 1.2: MIMD Architectures
shared is a prime factor in determining the coupling of a system. Note in Figure 1.2 that in the loosely-coupled case each processor has a private memory; in the tightly-coupled case the processors share a common memory.

A key feature of both types of MIMD architectures is the interconnection network. This network provides the physical mechanism for communication. Besides the usual processor/memory communication, communication is required between processor/processor pairs and from processors to any specialized resources in the system. The design of this communication network not only effects the performance of the network itself but has far-reaching influence on the performance of the entire system, not only of the hardware but also on the software used to drive the system. For this reason it is necessary to evaluate the performance versus cost tradeoffs of interconnection network design.

Interconnection Networks

The importance of communication in multiprocessor systems cannot be overestimated. Many parallel algorithms have shown poor performance not because the complexity of the algorithms is bad but because the interprocessor communications cost was too large and dwarfed the cost of other parts of the algorithms. Additionally, with the advent of inexpensive processors and memories not only must intermodule communication be efficient it must also be inexpensive so that the cost of the system is not dominated by that of the interconnection network. A current commercial processor which is optimized for multiprocessor architectures, Inmos' Transputer [9], is based on these concepts.

The purpose of an interconnection network is to provide a communication path from one of a set of sources to one (or several) of a set of destinations. Multiprocessor architectures can be divided into two categories based on their interconnection networks. These are processor-centered and network centered systems [10]. In the processor-centered case the performance of the processor is the main factor in determining system performance. The interconnection network is not a global resource. Hypercube architectures [11] are a good example of this class of systems. Conversely, in network-centered systems the interconnection network is viewed as a global system resource which must be shared by all processors. In this case network performance will have a much stronger influence on
system performance. Processor-centered systems are not considered further in this dissertation.

The specification of interconnection networks requires several design decisions to be made. In Feng's survey article [12] it is stated that the design of an interconnection network can be chosen from a set of designs described by the cross product of several parameters: \{operation mode\} × \{control strategy\} × \{switching strategy\} × \{network topology\}. In this dissertation all design parameters of the network will be fixed except for the network topology, and in Chapter 4, control strategy. The choices from the other categories and fixed for all of the networks under consideration are:

Operation mode -

The operating mode is selected from the set \{asynchronous, synchronous\} and is chosen to be asynchronous due to the MIMD structure of multiprocessors. The asynchronous mode means that connection requests can be issued at any time. This does not imply that the implementation of the network uses asynchronous logic, only that requests originating at one processor may be issued asynchronously of requests originating at other processors.

Control strategy -

The control of the network switches is distributed rather than centralized. In this scheme each of the switches in the network is responsible for directing messages appearing at its inputs to the appropriate switch output. A distributed algorithm alleviates the potential bottleneck of the single scheduler which performs these routing decisions in the centralized control strategy. Particular methods of performing distributed routing will be discussed in the later chapters.

Switching strategy -

Network switching can either be done as packet switching or as circuit switching. In packet switching each message to be transmitted is divided into multiple packets which are routed individually to a destination. There is no requirement that the packets arrive at the destination in a contiguous manner. Packets from other messages which have the same destination may be interspersed in the stream of packets arriving at the destination. In circuit switched networks the packets of a message must form a contiguous stream; no packets from other messages may break the stream of packets. All networks discussed here will be circuit switched. A distinction
should be made between switching methodologies and network buffering. Circuit-switching
does not imply that the network must be unbuffered nor does packet-switching imply buffered
networks. There are obvious reasons why unbuffered networks are not packet-switched but
Walkup [13] has shown circuit-switched buffered networks to be feasible. Packet-switched net-
works can be investigated as a special case of circuit-switched networks.

Topologies -

The networks considered in this section are restricted to three well defined types of networks.

All three topologies have single paths between all source/destination pairs. In the following
description of the three types the bandwidth of a network is the maximum number of packets
which can exit the network in a unit time interval.

(1) The crossbar has the largest bandwidth of the three networks under consideration. Figure 1.3
shows a $4 \times 4$ crossbar network. A crossbar is structured such that any source/destination con-
nection can be made regardless of the existing connections in the network, as long as no two
sources attempt to connect to a common destination. A network with this property is known as
nonblocking. Unfortunately, this flexibility is achieved at a price, the crossbar is also the most

---

![Diagram](image)

*Figure 1.3: $4 \times 4$ MCC Crossbar Network*
complex of the three networks and therefore the most expensive.

(2) The bus is the simplest of all networks as well as the cheapest. Figure 1.4 shows a bus network. All modules which need to communicate with other modules are interfaced to a common link. No switching is used to establish connections between source/destination pairs. A bus network is clearly a blocking network.

(3) The multistage network (MSN) is a compromise in performance and cost between the crossbar and the bus. An $8 \times 8$ MSN constructed from $2 \times 2$ crossbar switches is shown in Figure 1.5. Each column of switches is known as a stage of the network. There are several patterns that the connections between stages can have but the equivalence of the patterns has been shown in previous work. The MSN is of particular interest because it provides nearly all of the bandwidth of a crossbar while using fewer switches.

Delta networks, the type of network investigated in this dissertation, are multistage networks in which routing of messages is performed at each stage by inspecting a particular bit in the destination address associated with the message. This algorithm has been described by previous investigators and is as follows: assuming an $N \times N$ network where $N = 2^n$ and therefore destination addresses are $n$ bits. If the input stage is numbered stage 0 and the output stage is numbered stage $n-1$ then at the $i^{th}$ stage

![Bus Network Diagram](image)

*Figure 1.4: Bus Network*
the \((n-i)^{th}\) bit of the destination address is inspected; if the selected bit is a 0 the message is routed to the upper output port, if the bit is a 1 the message is routed to the lower output port. This method results in a simple distributed control algorithm for the network.

A link refers to the connection between an input port and an output port. An input link refers to a link which is driven by a message source and an output link refers to a link which routes data to a message sink. Because a delta network is a blocking network two messages arriving at different switch inputs may require routing to a common switch output. In this case a collision occurs and one of the messages must be delayed. In Figure 1.6 a delta network switch is shown. The buffers on the switch inputs are used to hold messages temporarily when the rate of arrivals exceeds the rate at which packets are transmitted. These buffers are of finite depth. Previous work by Jump and Dias [14] has
shown that optimal depths are 2 to 3. If the buffers fill network operation requiring the switch output cannot continue until a buffer position becomes free. More detail on the operation of delta networks modeled in this dissertation is presented in Chapter 3.

Problem Statement

The hypothesis investigated by this research is that in a large number of parallel architectures the use of a full multistage interconnection network such as a delta network is not cost-effective. Three arguments are presented in defense of this statement. First, if the service time required at the destination side of the network is moderate to large the bandwidth of the network cannot be utilized. In this situation the system bottleneck occurs at the destinations. Second, interconnection networks achieve their throughput rates because they form a pipeline for service requests. If the application using these pipelines cannot sustain a sufficient number of concurrent operations to keep the pipelines filled then bandwidth is wasted. Third, the practical problem of implementation must be considered. The I/O requirements of a full multistage network must be satisfied if peak throughput rates are to be obtained.

The dissertation investigates how much communication bandwidth is needed for a system and shows how a hybrid network, constructed by replacing stages of a delta network with buses, can be used to provide sufficient levels of performance at a reduced cost. Implementation of delta networks with respect to constraints imposed by VLSI is also considered. These constraints lead to another method of reducing network cost; the performance of this method is compared to the performance of
hybrid networks.

Dissertation Outline

The investigation of the problems stated in the previous section is organized into several chapters.

Chapter 2 -
presents a more detailed explanation of the problems presented in this chapter.

Chapter 3 -
develops an analytical model of delta and reduced bandwidth delta networks. Performance evaluation computed from this model is compared with a set of simulations performed on these networks. Differences between the model and the simulations are discussed. The development of a new model for the system load is presented and used to determine how much interconnection network bandwidth is required for a given computation.

Chapter 4 -
is concerned with the performance of hybrid networks when they are used as resource sharing networks. Resource sharing networks are used to connect specialized system resources to the processing modules in a multiprocessor system. A more detailed description of the network’s function and operation is presented in the chapter.

Chapter 5 -
investigates a problem encountered in the VLSI implementation of interconnection networks namely the pin limitation problem. The problem of network partitioning is discussed and an alternative method of reducing delta network bandwidth is presented.

Chapter 6 -
presents a summary of the results obtained during the investigation along with a discussion of several avenues of further research in the area.
CHAPTER 2

Motivation

Previous investigations into the performance of buffered multistage networks have been based on several assumptions:

(1) Arrivals to the network occur at discrete times with probability $p$ of an arrival occurring on any input link in a single time period. The arrival on one input link is independent of the arrival on any other input link.

(2) Destinations for arrivals are chosen according to a uniform probability distribution.

(3) Packets are removed from the network instantaneously upon arrival at their destination.

(4) Many investigations [15, 13] also assume an arrival probability of $p = 1$. The motivation for this assumption is the desire to investigate the heavy load performance of network as well as to establish the maximum throughput of the network.

Assumption (3) in the above list leads to a one-way model of interconnection networks. In this type of model consideration is only given to the flow of messages in a single direction. This can be a misleading model of an actual system for the following reasons. First, network communication is usually bidirectional. A message is sent to a destination with the prospect of receiving a reply which contains some information desired by the sender or at least an acknowledgement of the receipt of the message. Second, in general, the time required to formulate such a reply is non-negligible. This time, the reply time denoted by $R$, can have significant influence on the performance of the system. These two arguments lead to the proposal of an extended model for the investigation of interconnection networks.

System Model

The new model is changed in several ways. The first departure from the model described above is the addition of a reply network. This network transfers messages from the destination side of the
system to the source side. It operates identically to the request network except that traffic on the networks flow in opposite directions. To clarify in which direction a message is moving the term request is used to describe a message originating at the source side of the system and traveling to the destination side of the network. Requests are entered into the network as a result of the model used to describe the system load. The other type of message, a reply, is transmitted from the destination side of the network in response to the receipt of a request from a source. The source side of the network may be considered the processor side; the destination side of the network may be, but is not restricted to be, a memory system.

The second departure from previous results is to consider a delay incurred at the destination side of the network when a complete message is received. This models the time required to perform the requisite computation and to formulate a reply to the original message. For example, if the system under consideration was a processor/memory interconnection network then this delay might represent the time required to perform the actual read or write cycle within the memory. It is assumed that the generation of a reply cannot begin until a complete message has been received.

A third change involves the model of the arrival rate. For reasons seen more clearly in following sections emphasis is not on the model of a heavily loaded system. Instead, the arrival rate is varied as a parameter of the model. Later, another mechanism for the generation of arrivals is introduced. In some parts of this investigation the system will be considered closed. Processors will generate messages and wait until replies to the messages are received before generating subsequent messages. This model is discussed further when results generated by its use are presented.

Operating in a round-trip mode, this model more accurately reflects real systems and demonstrates the dependency of network performance on other parts of the system. Immediate evidence of this fact is that an \( N \) processor system must send messages through \( 2 \log N \) stages, \( \log N \) for both request and reply networks, rather than the \( \log N \) stages in previous models. At the very least, this fact increases the minimum message delay time by a factor of 2. It will be seen that other phenomena occur. Understanding the dependencies that exist will prove to be critical as new architectures utilize more complex interconnections than the bus networks which dominate current systems.
System Bottlenecks

A system bottleneck is defined to be the component of the system which limits performance as the system approaches its maximum performance state. By definition every system has a bottleneck although it may be the case that multiple system components act as bottlenecks simultaneously. One purpose of performance analysis is to determine where this bottleneck is and what to do about it, if anything. Previous investigations in the interconnection network area have made the implicit assumption that a system's bottleneck will be the network itself; otherwise the assumption of heavy loading often used in previous work makes little sense. The following arguments show that in many situations the network is far from being the bottleneck of the system. It is assumed in the remainder of the dissertation unless otherwise stated, that the network in question is a delta network. More details of network operation will be stated as required.

Performance Measures

In both the analytic models and the discrete-event simulation models two figures of merit are used to compare various systems' performance. They are network throughput and message delay. Both figures will be normalized where appropriate.

*Network throughput, TP,* is defined as the number of reply messages received per unit time. Note that the number of reply messages transmitted forms a one-to-one correspondence with the number of request messages transmitted minus the messages currently en route to the destination side of the network. For long simulation times this difference is negligible. *Normalized network throughput, NTP,* is calculated by dividing TP by the number of network outputs.

*Message delay, D,* is defined as the period of time which begins when a source offers a request to the network and ends when the source has completely received the reply corresponding to the request. The delay is normalized, ND, by dividing the observed values by the minimum time required to transit the request network, generate a reply, and then transit the reply network. Time is measured in units corresponding to the interstage transfer time of the network.
Another term associated with the performance of a network, *network bandwidth*, will also be used. Network bandwidth is defined as the maximum throughput which can be achieved by a given network. This is generally equal to the maximum number of parallel paths from network input to network output.

Large Reply Times

One situation where the entire bandwidth of the network is not used is when the time required to generate a reply is large relative to the time required to transfer a message between stages. It seems obvious that this would be the case. The interesting question is: to what extent does the reply time influence the performance of the network?

To begin consideration of this question it is desirable to see how the bandwidth of the new model is affected by the two system parameters, reply time, $R$, and arrival rate, $p$. A graph of this performance curve for a network of size 64 is shown in Figure 2.1. Let an *ideal network* be a network in which no collisions occur either at the switches or at the resources, *i.e.*, destination addresses are uniformly distributed and properly sequenced in time so that no two packets require the same link at the same time. It is seen that the bandwidth of the ideal system is bounded by $\frac{bw}{R}$, where $bw$ is the bandwidth of the network.

Figure 2.1 demonstrates that in an ideal network a mismatch between the speed of the network and the reply time causes a decrease in network throughput proportional to the reply time. For example, if a network is used as a processor/memory interconnection and the time to transfer between stages of the network is 60ns, a reasonably fast network, and the rate at which a memory bank can be accessed is 240ns, reasonable considering the need to perform error correction and detection on data stored in dynamic rams, then at heavy loads the throughput of the network is decreased by a factor of 4 as compared to its bandwidth.

Applications involving values of $R$ greater than 10 is quite possible as use of interconnection networks more sophisticated than buses becomes more common. Jump et al. [16] propose to perform high-level vector operations (HLVOs) in auxiliary programmable functional units. The HLVOs, in
general, may be quite complex and may involve data dependent branching and floating-point operations. It is not difficult to imagine operations requiring several microseconds to compute even on very high-speed functional units. This time translates into values for $R$ of several hundred. Others [17, 18, 19] have also proposed large-grain parallelism as a method of increasing system performance. Use of parallel interconnection networks must be considered in systems where communication is not restricted to conventional processor/memory networks. It is necessary to consider the use of interconnection networks in a larger area of application than has historically been done.

**Insufficient Concurrency**

Another scenario in which the entire bandwidth of a multistage network is not utilized is the lightly loaded case. If the sources of traffic for the network are not using the services provided at the destination side of the network frequently the cost of a fast interconnection network is not justified.
This situation is not likely to occur in situations where the network is acting as a processor/memory connection, although with the increasing use of both instruction and data caches integrated with a single chip processor, the traffic between the processors and memories will be less than in cacheless systems. However, if the network is used to interconnect processors and specialized functional units in a tightly-coupled system such as proposed by Thomason and Avizienis [20], Briggs et al. [19], and Jump et al. [16], then consideration of this problem is reasonable. More general use of interconnection networks may also be found in dataflow architectures.

Implementation Problems

As technology associated with VLSI devices progresses more sophisticated systems are being integrated into single packages. Current commercial CPU chips have upwards of half a million transistors each and dynamic RAM packages, the leading edge of device density, are an order of magnitude beyond that. Yet there has been little effort made to integrate an interconnection network into a single package solution. This is clearly not due to the inability to place a sufficient number of devices in a package. The problem lies in another area: severe limitations on I/O connections. Current technology does not provide enough pins on a package to allow parallel data paths for large numbers of processors.

For instance consider the implementation of a square delta network (i.e., a network with an equal number of inputs and outputs) on a single chip. Let \( N \) be the number of inputs to the network and \( W \) be the width of a data path. The number of I/O connections required is \( C = 2NW \). The state of today's technology in pin grid array (PGA) packages allows approximately 625 (square grid, 25x25) pins. A value for \( N \) of 256 is considered to be of moderate size; for a moderate sized delta network to be implemented in a single package the data paths are constrained to be only a single bit wide. Several options are available to solve the pin limitation problem. Two of these, the use of networks with narrow data paths and the use of multiple packages operating in a parallel, bit-sliced manner, are considered.
Summary and Proposals

In the previous sections of this chapter arguments have been presented for the following points:

(1) When discussing the advantages and disadvantages of interconnection networks the performance of the entire system must be considered; restriction of investigations to the performance of a network in isolation possibly leads to misleading evaluations of system performance. Incorporating the effects of the system as a whole into a model by considering both the necessity of a reply message and a non-zero reply time showed that at relatively small values of \( R \) the system bottleneck is the rate at which service can be supplied by the destination side of the network. This is in contrast to the conventional model of interconnection networks which assumes that the bottleneck will be the network.

(2) Technological constraints must be considered. Current proposals for multiprocessors envision large (> 1024) numbers of processors. At this time fully parallel interconnection networks for these systems cannot be implemented on a single chip due to I/O restrictions. The performance of various types of network implementations for these systems needs to be investigated.

The investigation begins by developing a model for the generalized delta network. Performance results obtained from this model suggest that available network bandwidth is underutilized for the reasons mentioned previously. A solution to this problem is proposed and analyzed. In the second section of the dissertation, the implementation of delta networks is considered and results demonstrate a second method for reducing network bandwidth. Finally, the comparison of these two methods links the two problems, network cost-effectiveness and network implementation, together.
CHAPTER 3

Address Routed Networks

The arguments stated in Chapter 2 point to the need for an interconnection network which provides levels of cost and performance which lie between those of the full multistage network and the bus. This hints at a larger problem. Given a computation to be performed on a multiprocessor such as was described in the previous chapters, how much interconnection bandwidth is needed to allow the computation to be performed in the minimum possible time? This question can be rephrased. How slow can an interconnection network be and still not be the system bottleneck? This is an important question because the cost of complex networks is so high.

This chapter begins by developing an analytical model of a delta network. Results from this model indicate that there is a need for networks with cost/performance levels between that of a bus and a delta network. A set of networks is proposed to fill this need and their performance is considered. Finally, results are presented which indicate how much bandwidth is needed in the system interconnection network for a given system load.

Models

In this section of the thesis a numerical model of a buffered delta network is developed and applied to hybrid networks. Previous researchers in this area have taken different approaches to the development of models for multistage networks. Among them are Dias [15] and Walkup [13] who independently modeled the system by using Markov chains with the state of an individual switch representing a state in the chain. They applied these methods only to the case where a single buffer was present at each switch. They note that for more complex switches, the number of states becomes very large and prevents analysis with a reasonable amount of effort. Kruskal and Snir [21] model a switch as an M/G/1 queuing network. They show this to be a reasonable approximation to results they obtain by simulation. It is also noted that their analysis cannot be extended in a recursive manner over
the stages of an entire network due to correlation between arrivals at stages other than the first stage. This problem was noted by Dias and Kumar [22] in response to results presented by Thanawastien and Nelson [23] who recursively applied a Markov chain solution of the first stage to successive stages of the network. Kruskal et al. [24] also use an approach based in queuing theory to formulate several approximations of network behavior (particularly waiting time distributions) in latter stages of buffered multistage networks.

The model developed in this section is not intended to be exact. The increased complexity of today's systems combined with the growing need to model subtle details of operation are forcing simulation to be used as the primary tool for performance evaluation. Additionally, high quality simulation packages, such as the RESQ package developed by IBM and the CSIM package [25] developed at Rice University, are readily available so simulations can be developed easily. The analytic model then serves as a quality assurance for simulation correctness as well as a quick, less computationally expensive method of providing rough performance measures.

The inherent regularity of delta networks suggests that a network model could be constructed by aggregating several instances of a model of a single switch or stage of switches. This is not a new idea, Patel [26] and others [27,23] have used a recursive approach to find the blocking probability in unbuffered delta networks.

Model development begins by specifying and solving a queuing network approximation of a 2 × 2 crossbar switch. This is the building block which is aggregated. Next, because results pertaining to the effects of large resource service times are of interest, a model of the resources is developed. Third, the single switch model and the resource model are combined to form the network model. An important consideration in this step is how the dependencies between stages are represented. Finally, the network model is evaluated and performance measures are discussed.

Switch Model

The queuing model for a single switch is shown in Figure 1.3. The switch has finite length buffers of depth \( k \), the service requirement is deterministic (a single time unit), and the service discip-
line is FCFS. The state of the model is determined by the number of packets in the system. To decouple the two halves of the switch an assumption is made that the limiting state probabilities of each queue are independent of the state of the other queue. This independence is not found in a physical network; the state of one buffer affects the state of the other buffer. The assumption of independence is made to aid analysis. Let $\pi(i_0, i_1)$ be the steady state probability that there are $i_0$ packets in queue 0 and $i_1$ packets in queue 1 and let $\pi(i)$ represent the steady state probability of either half of the switch being in state $i$. The assumption of independence allows the joint state probabilities of the switch to be written as the product of the individual queues' state probabilities.

$$\pi(i,j) = \pi(i) \pi(j)$$

(3.1)

From this point, all state probabilities and state transitions will refer to the operation of a single half of the switch.

Arrivals to the system are independent and occur at each input with probability $p$ during each time period when the system state is less than $k$; when the system is in state $k$ arrivals occur with probability zero. Blocking of a packet due to contention for an output link is modeled by the feedback path. After a unit time delay, the packet at the head of the input queue takes this feedback path with probability $b$. Because the standard assumption of uniform destination routing is made, taking the feedback path back to the queue is equivalent to remaining in service for another time interval. If $b$ is
known then the model is solved trivially.

Since an arriving packet must remain in the switch for at least one time unit, no blocking can occur unless both queues are non-empty. In this case a collision occurs if the packets are both routed to the same output. This occurs with probability $\frac{1}{2}$. If a collision occurs, one packet is blocked and the other is transmitted. Since each is chosen for transmission with equal probability, the blocking probability for a packet ready to be routed is $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$ if there is also a packet in the switch’s other buffer. The probability that a packet is routed back to the queue through the feedback path is given by equation (3.2).

$$b = \frac{1}{4} \left[ 1 - \pi(0) \right]^2$$

To find the value of $b$ it is necessary to find the values for the limiting state probabilities, $\pi$. These probabilities form a $(k+1)$-element vector and can be found by manipulating the transition matrix, $T_b$, of the Markov chain which describes the queuing model. The state transition matrix for $k = 5$ is shown in Figure 3.2, $\overline{p} = 1 - p$ and $\overline{b} = 1 - b$. All elements of the transition matrix are zero except for the $(i, i-1), (i, i)$, and $(i, i+1)^{th}$ elements of the matrix, $0 \leq i \leq k$, which are given by equation (3.3).

$$T_5 = \begin{bmatrix}
\overline{p} & p & 0 & 0 & 0 \\
\overline{p} \overline{b} & p \overline{b} + \overline{p} b & p b & 0 & 0 \\
0 & \overline{p} \overline{b} & p \overline{b} + \overline{p} b & p b & 0 \\
0 & 0 & \overline{p} b & p b + \overline{p} b & p b \\
0 & 0 & 0 & \overline{b} & b
\end{bmatrix}$$

Figure 3.2: Transition matrix for $k = 5$
\[ T_k(i, i-1) = \left(1 - b\right) \left(1 - p\right) \]
\[ T_k(i, i) = p \left(1 - b\right) + b \left(1 - p\right) \]
\[ T_k(i, i+1) = pb \] 
(3.3)

Forcing the arrival rate to zero when the system is in state \( k \) makes the implicit assumption that packets arriving at that time are lost.

Unfortunately, the transition matrix, \( T_k \), and equation (3.2) have coupled \( \pi \) and \( b \). However, the limiting state probability vector can be approximated by multiplying \( T_k \) by itself several times [28].

\[ \pi(j) = \lim_{n \to \infty} T_k^n(i, j) \] 
(3.4)

The actual value of \( n \) required depends upon the desired accuracy of \( \pi \). Details of the convergence of \( T_k^n \) to \( \pi \) are given by Cinlar [28].

The process of determining \( b \) consists of guessing an initial value for \( b \) and applying equation (3.4) to determine \( \pi \). The calculated value of \( \pi(0) \) is then substituted into equation (3.2) to determine a blocking probability \( b' \). If the difference between \( b \) and \( b' \) is sufficiently small then the blocking probability has been found. If the difference has not converged then a new value of \( b \) is chosen and the process is repeated. Several techniques can be used to select the new value of \( b \). Simple substitution of the average of the current guess and the calculated \( b' \) for the new guess works well; typically fewer than 5 iterations are required to achieve convergence.

Figure 3.3 shows the blocking probability as a function of arrival probability for a switch using buffers of depth 2, 5, and 10. A very interesting observation is made from this graph. At intermediate arrival rates the blocking probability is higher for larger buffer depths. While at first consideration this seems contradictory, it is easily explained. The blocking probability varies inversely with the probability that the buffers are unoccupied. Since larger length buffers tend to be occupied more frequently the blocking probability will be correspondingly higher.

Now that \( b \) is known other measures can be made. For instance, expected queue length, \( \bar{Q}_k(p) \) can be found directly from \( \pi \).
\[ \overline{Q}_k(p) = \sum_{i=1}^{k} i \pi(i) \]  

(3.5)

Figure 3.4 shows expected queue length as a function of arrival probability for buffer lengths of 2, 5, and 10.

The process of passing a message through the switch can be considered a Bernoulli trial with a success being indicated by the message leaving the switch and a failure being the event that the message blocks. Using this analogy, \( \overline{N}_k \) is the expected number of failures before a success occurs and is calculated by equation (3.6). Equations (3.5) and (3.6) are of importance because their product gives the expected waiting time for a message in the switch.
\[ \bar{N}_b = \sum_{i=0}^{\infty} i b^i (1 - b) \]
\[ = \frac{b}{1 - b} \]  

(3.6)

Resource Model

The solution of the model for a single switch is complete. Before developing the model for an entire network, a model of the network resources must be considered. Recall that the system being investigated has non-zero service times at the resources. This causes the number of resources present at each network output and their individual performance to have significant effects upon the performance of the system as a whole. In the conventional delta network model there is only one resource per output. The model discussed here allows multiple resources per output so that results can be used in a later section.
The model of a resource in this investigation is fairly simple. There is no buffering at the resources, a single class of service is provided, and the service time distribution is geometric with mean $R$, i.e., the service process has a parameter $r = \frac{1}{R}$. Each network output link is connected to $S$ resources. Arrivals occur with probability $p$ during each unit time interval and select resource $i$, $0 \leq i \leq S$, with probability $\frac{1}{S}$. If the resource selected is idle it accepts the packet and begins operation. If the resource is busy the packet waits until the resource becomes idle. During the time a packet is blocked no other arrivals may occur. Figure 3.5 shows the resource model.

Let $(q_0, q_1)$ be the current state of the system and $(q'_0, q'_1)$ be the next state of the system. The first element of the state vector, $q_0$, represents the number of resources currently busy. The second element, $q_1$, is binary valued and indicates whether or not the system is blocked. The behavior of this system can be represented by set of state transitions whose probabilities are given by expressions (3.7-3.10). \( \binom{n}{r} \) represents the number of combinations which can be formed from $n$ things taken $r$ at

\[ \binom{n}{r} = \frac{n!}{r!(n-r)!} \]

Figure 3.5: Resource System Model
a time, and \( Pr \left( (q_0, q_1) \rightarrow (q'_0, q'_1) \right) \) is the probability of a transition from state \((q_0, q_1)\) to state \((q'_0, q'_1)\).

\[
Pr \left( (q_0, 0) \rightarrow (q'_0, 0) \right) = (1-p) \begin{pmatrix} q_0 \\ q'_0 \end{pmatrix} (1-r)^{q'_0} \frac{q_0}{q'_0} \left( 1 - \frac{S - (q'_0 - 1)}{S} \right) \left( 1 - (q'_0 - 1)^{q_0 - q'_0} \right)
\]

\[
Pr \left( (q_0, 0) \rightarrow (q'_0, 1) \right) = p \begin{pmatrix} q_0 \\ q'_0 \end{pmatrix} (1-r)^{q'_0} \left( 1 - \frac{S - (q'_0 - 1)}{S} \right) \left( 1 - (q'_0 - 1)^{q_0 - q'_0} \right)
\]

\[
Pr \left( (q_0, 1) \rightarrow (q'_0, 0) \right) = r \begin{pmatrix} q_0 - 1 \\ q'_0 - 1 \end{pmatrix} (1-r)^{q'_0 - 1} (q'_0 - 1)^{q_0 - q'_0 - q'_0}
\]

\[
Pr \left( (q_0, 1) \rightarrow (q'_0, 1) \right) = (1-r) \begin{pmatrix} q_0 - 1 \\ q'_0 - 1 \end{pmatrix} (1-r)^{q'_0 - 1} (q'_0 - 1)^{q_0 - q'_0 - q'_0}
\]

Using equation (3.4) and the transition matrix determined by (3.7-3.10), the limiting state probabilities of the system, \( \pi_{\text{resource}} \), can be found. Equation (3.11) calculates the blocking probability of the resource subsystem; expected utilization, \( U \), and the expected throughput, \( NTP \), of the resources are given by equation (3.12).

\[
Pr_{\text{resource system blocked}} = \sum_{i=0}^{S} \pi_{\text{resource}}(i, 1)
\]

\[
U = \frac{\sum_{i=0}^{S} \left[ \pi_{\text{resource}}(i, 0) + \pi_{\text{resource}}(i, 1) \right]}{S}
\]

\[
NTP = \frac{U}{R}
\]

Network Model

Having developed the model of a single switch and the model of the resource subsystem efforts are now directed towards the modeling of an entire network composed of a collection of these models. The difficulty arising in this is that departures from a switch are no longer only dependent upon the state of the switch but now also have dependencies on other network stages. Clearly if a switch in the
ith stage has full buffers then it cannot receive messages from switches in the \((i-1)th\) stage. This type of dependence exists throughout the network. The final stage encounters this problem when resources at the destination side of the network are not available to provide service.

One method of modeling the problem described above is to modify the blocking probability of the individual switch model. Blocking is a result of two factors. First, blocking can be caused by contention between ports of the same switch for a common output link, \(b\) models this factor. Second, blocking can be caused by the unavailability of buffer space in the switch to which the current message is being passed.

\[ P_{\text{message blocks}} = P_{\text{intra-switch contention for link}} + P_{\text{subsequent stage cannot accept message}} \]  

(3.13)
The second factor can be difficult to characterize for general depths of buffering. The majority of network models developed to this date use one of two buffering schemes: infinite buffering, which eliminates blocking due to full buffers, and single buffering, which allows a small number of states to characterize the network. The first approach is far from realistic under most circumstances, the second is useful but not as general as is desired.

Some new notation is introduced at this point, \(b\) and \(\pi\) continue to represent the blocking probability and the limiting state probabilities of a switch in isolation. In the network model, \(b_s, p_s, \text{ and } \pi_s\) represent the blocking probability, limiting state probabilities, and arrival probabilities of the \(s\)th stage. The implication is that these quantities, in general, are different for each stage. The depth of the buffering at each stage is \(k\).

To determine what modifications should be made to the blocking probability of a switch when it is incorporated into the network model, consider Figure 3.6. A message from stage 1 to stage 2 is blocked when two conditions hold:

1. A message must be sent from stage 1 to stage 2. Obviously, if no message is sent, none can block.
2. Stage 2 must be in state \(k\). If it is not in state \(k\) then there is a free buffer and blocking cannot occur because of insufficient buffers.
From the preceding arguments and equation (3.13), equation (3.14) expresses the blocking probability of a switch when it is subjected to the effects of other stages. Note that all blocking and state probabilities are functions of the network arrival rate. This dependence is not shown explicitly in order to keep the notation compact.

\[ b_1 = b + (1 - b) \pi_2(k)|_{p_2} \]  

(3.14)

The bar adjacent to \( \pi_2(k) \) indicates that the function must be evaluated at \( p_2 \). Recall that \( b \) and \( \pi(k) \) are functions of arrival probabilities. The need to evaluate the function at \( p_2 \) presents a difficulty as \( p_2 \) is not known. Equation (3.15) is used to determine \( p_2 \).

\[ p_2 = \left[ 1 - b_1 \right] \left[ 1 - \pi_1(0) \right] \]  

(3.15)

Another problem has now arisen. The value of \( b_1 \) is used to determine \( p_2 \) which in turn is used to determine \( b_1 \). Iterative techniques must be used to find consistent parameters for the system. The iteration works as follows:

1. Pick an initial guess for \( b_1 \).
2. Using the state transition matrix described in the single switch model, determine \( \pi_1 \). Use these results to calculate \( p_2 \) from equation (3.15).
3. Use the calculated value of \( p_2 \) and equation (3.14) to calculate \( b'_1 \).
4. Compare \( b_1 \) with \( b'_1 \). If the difference is sufficiently small then \( b_1 \) is valid. Otherwise replace the old value of \( b_1 \) with the value of \( b'_1 \) and repeat from step (2).
To solve the network model the iteration procedure must first be applied to the combination of the final stage and the resource subsystem. Note the resources replace stage 2 in Figure 3.6 except that in place of \( \pi_2(k) \), \( Pr_{\text{resource system blocking}} \), equation (3.11), should be used. In this manner the blocking of the final stage due to resource unavailability is reflected as an increase in \( b_{n-1} \). Once the adjustment has been made to the final stage the procedure is applied to the combination of the next to last stage and the adjusted values of the last stage. The procedure is repeated until adjustments are made to the blocking probabilities of the input stage.

It may seem that the computation involved in solving the model would be excessive. However, experience gained in developing the model and in forming comparisons with simulation results of the same system shows that for moderate buffer depths the computation time required to solve the model is much smaller than the time required to simulate the system.

The end result of this set of iterative procedures is a set of state probability functions for each stage in the network. These equations describe the network state which is seen by an arrival to the network. Equation (3.15) can be used to recursively determine the arrival rate at the resources. At that point equation (3.12) can be used to calculate the NTP of the network.

Model Verification

Now that the development of the model and the mechanism for its use has been presented it is necessary to determine how closely the performance predicted by the model correlates with the performance predicted through the use of simulation. At this time only full delta networks with a single server at each network output are considered. Within this class of networks several parameters can be varied. The following sections use a fixed set of parameters, 6 stages, expected service time of 2.0, and a buffer depth of 2, and then vary one parameter to see how the model and simulations alter their performance predictions in response to the changes in each parameter.

Buffer Depth -

One of the parameters of the model used in this dissertation which has not been considered in a general manner by prior investigators is the depth of the buffers at each switch. Figure 3.7
shows NTP levels as a function of arrival probability as predicted by the model and a simulation. The plots are parameterized by the value of $k$.

Resource Service Time -

The other innovation of this model is to include the effects of non-zero reply times on system performance. Figure 3.8 compares NTP levels for the simulation and the model when $R$ is varied. The model's accuracy is fairly insensitive to the value of $R$.

Model Inaccuracies

The previous section showed that while the model produces reasonable approximations of the performance calculated by the simulations, there is some error present. This section discusses some potential origins of this error. The sources fall into two general categories, error caused by the model failing to accurately represent the system under consideration, and error caused by solving the model

---

![Graph](image)

Figure 3.7: NTP vs. $p$: $N = 64$, $R = 1.0$
incorrectly.

Dependence Between Switch Queue States -

In the section describing the single switch model, an assumption is made that the states of the buffers at each switch input are independent. This is used to allow the joint state probability to be expressed as the product of the individual state probabilities. The state probabilities are not independent and therefore the model will not precisely reflect the behavior of the network.

Single Switch State Probabilities -

Limiting state probabilities for the single switch model were found by multiplying a state transition matrix by itself several times. This technique is an approximation to an identity stated in Cinlar [28]. The exact solution is given by the matrix to raised to the \( \infty \) power; clearly an exact solution cannot be found using this method to find the state probabilities. However, this technique is convenient and fairly inexpensive and the error introduced is not large.
Convergence of Network Model -

The effects of messages blocking due to full buffers in the subsequent stage is modeled by increasing the blocking probability of the current stage. This requires an iterative technique to be used so that two constraints are satisfied. The iteration causes parameters to converge only to within predetermined distances; the convergence is not exact. Additionally, some points do not even converge to the selected distance and thus must be approximated by interpolation techniques.

Discrete vs. Continuous Functions -

This is the largest cause of numerical error. Although the blocking probabilities and state probabilities are continuous functions they are only evaluated at a finite set of points. To find the solution for the network model, the evaluation of state probabilities for other stages is required. These evaluations are inexact if the evaluation point is not a point that was calculated previously. The error due to this phenomenon can be reduced as much as is required by simply evaluating the functions at a larger number of points. The tradeoff is, of course, the time required to solve and evaluate the model.

Hybrid Networks

The model presented in the previous section generates accurate predictions of system performance in a full delta network. This section shows how the model can be modified to solve the hybrid network problem. In the following discussion some additional notation is introduced. $N$ is the number of inputs to the network, $n = \log_2 N$ is the number of stages in a full delta network with $N$ inputs, and $n'$ is the number of stages removed from the full delta network to create a hybrid network.

The differences between the delta network and the hybrid network are found in the resource subsystem and the final stage of the network. Figure 3.9 shows a $16 \times 16$ hybrid network which has two stages removed. In this example $N = 16$, $n = 4$, and $n' = 2$. The changes to the model are:

1) multiple resources in each resource subsystem and
messages at output switches which are connected to the same resource subsystem always contend for a common link (the bus interconnecting the resources).

The model of the resources presented earlier is flexible enough to model the first change but the second change forces the original model of the single switch to be altered when the switch is in the final stage of a hybrid network. For the output link blocking now consists of two terms. The first represents contention within the switch; this is given by $b$ as developed earlier. The second term represents contention among the $S = 2^n$ switch outputs for access to a resource bus. Equation (3.17) is analogous to equation (3.2) developed earlier.
\[ b_{\text{final stage}} = b + (1 - b) \sum_{i=0}^{S-1} \frac{i}{i+1} \left( \frac{S-1}{i} \right) \left( 1 - b \right) \left( 1 - \pi(0) \right)^i \left( \pi(0) + b \left( 1 - \pi(0) \right) (S-1)^{-i} \right) \]

(3.17)

Using equation (3.17) as a constraint for the final stage of the hybrid network rather than equation (3.2) allows the model developed earlier to be applied to hybrid networks. The procedures to solve and evaluate the model are the same as those used for the delta network model except that the utilization of the resource subsystem must be evaluated for an arrival rate of \( S \) times the rate at each network output link because each of \( S \) switch outputs are contributing to the arrival rate at the resource bus.

An implicit assumption has been made in this model of hybrid networks with regard to the modeling of arbitration delay between the final stage of the network and the resources. The assumption made is that this delay is equivalent to the interstage delay. Detailed network design must be done to prove or disprove this assumption but as \( n' \) increases a point will be reached where the arbitration between all output ports cannot be performed in parallel. Therefore, delay will eventually be logarithmic in the number of ports being arbitrated. The base of the logarithm is highly dependent upon the actual implementation chosen. If the base is relatively large the approximation of a unit bus delay cost is reasonable and comparisons of a numerical model using unit bus delay agrees well with simulation results as is seen in Figure 3.10.

Modeling a bus delay greater than one is difficult to do without using simulation. This is because the deterministic nature of the bus requires that additional system state information be considered when performing state transitions. Rather than attempt to build an unwieldy and relatively inaccurate model of this system the results discussed in the remainder of this chapter will be obtained through the use of simulation techniques.

The simulation model used from this point on incorporates several changes to more closely model systems likely to be constructed. First, a round-trip model is used to model remote service provided to the processors. The reply network is assumed to operate in an identical manner to the request
network with the exception that replies received at the network sources are removed from the network in unit time. Second, no messages are dropped. If a message cannot enter the network on the cycle in which it is generated it is offered again in the next cycle. This process repeats until the message enters the network. Third, a bus delay model is used in which the bus delay is equal to $\log_2 S$. The final change to the model is the use of deterministic delays at the servers. $R$ now represents a constant resource service time.

Hybrid Network Performance

The purpose of hybrid networks is to provide a system interconnection with a lower cost than that of a delta network. Because hybrid networks do not provide the full set of connections available with the delta network it is expected that the performance of the hybrid networks will be lower than the performance of delta networks. To determine the degree to which the performance loss occurs the
NTP levels of hybrid networks are discussed. The NTP of an ideal delta network, a delta network which has ideal utilization of its links and resources, can be reduced below the network bandwidth for two reasons. First, the arrival rate to the inputs of the network may be low enough that the network is not saturated. This condition will be referred to as load limited. Second, the resources in the system may not be able to provide service at a rate high enough to prevent requests from queuing. This condition will be referred to as resource limited. It is clear that if \( p < \frac{1}{R} \) then the ideal system is load limited, otherwise it is resource limited. If the system is resource limited then expensive interconnection bandwidth is not being utilized. In this situation a hybrid network can be used in place of the delta network to reduce system cost without sacrificing system performance.

An ideal hybrid network, on the other hand, has three potential bottlenecks. The first two are the same as for the delta network. The third potential limitation is caused by the logarithmic delay of the bus. This condition will be referred to as bus limited. The network is bus limited when two conditions hold, \( p > \frac{1}{n'S} \) and \( R < n'S \). Recall that \( S \) is the number of network links contending for a common bus to the resources. A load limited situation occurs whenever \( p < \frac{1}{n'S} \); the network is resource limited if \( p > \frac{1}{n'S} \) and \( R > n'S \). Notice that the value of \( p \) determines only if the system has reached its limit of performance. It does not determine whether the network bottleneck is the bus or the resources. The bottleneck is determined by the relationship between \( n' \) and \( R \). Figure 3.11 presents the NTP vs. \( p \) graph of an ideal hybrid network. The next sections assume that the system is not load limited and use \( p = 1.0 \) to investigate the bus and resource limited conditions.

The bus is the network bottleneck whenever the NTP of the bus is less than the NTP of the resources. Equation (3.18) calculates the minimum value of \( R \) which causes the resources to be the limiting factor in system throughput.
\[ NTP_{bus} = \frac{1}{n'} \]
\[ NTP_{resources} = \frac{S}{R} = \frac{2^n}{R} \]
\[ R_{bottleneck} = n' 2^n \]  

It is easy to see that if the network is bus limited then additional performance can be gained by decreasing \( n' \) and if the system is resource limited the system cost may be reduced by increasing \( n' \).

Consider Figure 3.12 which shows the \( NTP \) of an ideal hybrid network as a function of resource service time, \( R \). The \( NTP \) is constant at its maximum value until \( R \) becomes the bottleneck. At that point the \( NTP \) decreases as a function of \( \frac{1}{R} \).

Unfortunately, equation (3.18) is true only for an ideal system. In non-ideal systems the situation is very complicated. Collisions caused by referencing busy resources will cause the bus to block even though there may be resources available to service other requests. Collisions in the request network affect the rate of arrivals at the bus and collisions in the reply network can cause resources to be underutilized. Deviations from ideal behavior can be seen in Figure 3.13 which shows the performance of hybrid networks connecting 64 processors to 64 resources. The four curves shown represent the performance of four systems. \( n' = 0 \) is the system with the highest \( NTP \) values, \( n' = 1 \) and \( n' = 2 \) lie
in the middle, and the performance for $\kappa' = 3$ is shown by the bottom curve. The dotted line is the graph of $\frac{1}{R}$ the asymptotic value for all systems.

Figure 3.14 isolates the curve for $\kappa' = 2$. The curve consists of a region, $R < R'$, in which the $NTP$ is constant and a region in which the $NTP$ varies inversely with $R$. The simulation shows the $NTP$ falls short of ideal levels in two respects.

1. $R'$ is significantly less than $\kappa' S$, the point at which the ideal $NTP$ curve changes from constant to $\frac{1}{R'}$.

2. The maximum $NTP$ value, $B'$ is substantially lower than the ideal value of $\frac{1}{\kappa' 2^{n'}}$.

It is desirable to modify the network such that the breakpoint of Figure 3.14, $B'$, is moved towards the ideal breakpoint, $B$. To do this the causes of the performance degradation must be identified. The shift of $B$ horizontally is caused by the inability of the resources to achieve maximum utilization; this is caused by transient non-uniformities in addressing patterns. As was mentioned above if two consecutive requests are directed to the same resource then the second request is not able
Figure 3.13: NTP vs. R
to be serviced and remains on the bus, preventing other requests from reaching their destinations. A straight-forward solution to this problem is to provide buffers at the resources. To totally eliminate effects of addressing non-uniformities infinite buffers must be employed. Fortunately, as shown in Figure 3.15, most of the degradation can be eliminated with a finite, fairly small number of buffers. The number of buffers used in Figure 3.15 ranges from 0, which produces the lowest performance curve, to 10, which produces the highest performance curve. The middle curves are generated by \( K = 1 \) and 2. The dotted line shows the level of performance in an ideal system.

The vertical shift of \( B \) is caused by the inability of the bus to operate at the maximum rate due to collisions in the network switches. Again, buffering can be used to increase performance levels. To shift \( B' \) upwards buffers must be added to the switches. The result of this buffering is shown in Figure 3.16. By adding buffers in these two ways the NTP of the non-ideal hybrid network can be made to
approach that of an ideal system. The buffer depths used in Figure 3.16 are \( k = 1, 2, \) and 5. Again the dotted curve measures ideal system performance.

This section has shown that hybrid networks can provide performance levels which fall between that of a bus interconnection and that of a delta network. The factors which determine network performance were discussed and it was seen that unless buffering is used both at the resources and within the network, system performance falls below ideal levels.

**Delay Driven Simulation**

The analysis of networks performed in the previous sections moves closer to actual network operation by including effects of system requirements such as round-trip operation and non-zero reply times. The performance of a given system was investigated as the load presented to it was varied. In this manner the architectural factors affecting system performance were determined. However, the
system load model was strictly probabilistic which is not always a good model of program behavior. In this section the investigation is centered around a new model of the system load.

Consider the operation of a processor which receives service from some resources. The service is obtained by submitting a request to the interconnection network and after some delay the results of the service are returned to the processor. This cycle, and therefore the rate at which the processor can compute, is limited by the delay between request submission and reply reception. This mode of operation forms the basis of a delay driven simulation model.

In general, a processor may not be restricted to only having a single service request outstanding. The ability to operate with multiple requests pipelined in the network may arise from multiprocessing abilities of the processor, in which case each process may have a request being serviced, or from the presence of independent operations within a single process. The delay driven model allows each pro-
cessor to maintain a maximum number of requests unresolved, $O_{\text{max}}$. Once a processor has submitted $O_{\text{max}}$ requests it blocks and waits until a reply to one of the requests is received. Then the processor is free to submit another request.

Actual processor behavior is best modeled by a combination of the probabilistic and delay driven models. The model considered in the remainder of this section operates as follows. First, an exponentially distributed delay is encountered by the processor. This models local computation. When the delay is complete and if the number of outstanding requests is less than $O_{\text{max}}$, a request for remote service is submitted to the network. The processor then waits until the number of requests outstanding is less than $O_{\text{max}}$ when the cycle begins again. Note that this model is a generalization of both of the other models.

Aside from more accurately representing the behavior of a real load, the new load model allows performance evaluation of the system to play a new role. Consider the operation of the system using the new load model. Initially, each processor delays some amount of time, submits a request for remote service and then delays some more, and submits another request. This process repeats until some maximum number of requests are circulating in the system. At this point, the processor submits a request only after a reply is received and the system remains in a dynamic equilibrium. In this state since requests are being offered to the network at a steady rate it seems reasonable to suppose that the same network performance profile could be created by using a strictly probabilistic load model with some arrival probability $\hat{\rho}$. In fact this is true. Plots of $NTP$ vs. $O_{\text{max}}$ show very similar characteristics to plots of $NTP$ vs. $p$. Increases in $O_{\text{max}}$ yield system behavior similar to increases in $p$. Noting this why should the more complex model of the load be used?

The reason for using the more complex model is that it decouples the model of the load from the model of the architecture. Consider the situation described in the preceding paragraph. Suppose for a given system and a given $O_{\text{max}}$, a probabilistic arrival rate of $\hat{\rho}_1$ is found to generate equivalent performance measures from the system. Now suppose that $O_{\text{max}}$ is held constant but a parameter of the architecture, for example the bus delay, is changed and new performance measures are made. In general, a probabilistic load of $\hat{\rho}_1$ submitted to the altered system no longer generates performance
measures which are equivalent to those generated by the delay driven model. Instead, a different rate, $\beta_2$, will generate a matching profile. The implication of this is that by using the delay driven model of the load, which is determined independently of the parameters of the architecture, it is possible to determine an architecture which is optimized for a given problem. In particular, given a characterization of the load in terms of two parameters, the number of requests allowed to be unresolved, $O_{\text{max}}$, and the amount of local computation performed between requests, $p$, the amount of interconnection network bandwidth required to support the computation can be determined. The new load model allows performance evaluation to move from a role of providing architectural analysis to one of providing architectural synthesis.

Architectural Evaluation Using a Delay Driven Load Model

Having described the advantages of the proposed load model discussion now turns to system behavior under the new model. It was noted earlier that considering network performance under the model does not yield any additional information over that which was obtained from the probabilistic model. Instead of directly considering performance measures of the network this analysis will focus on processor utilization. In the following investigations, characteristics of ideal networks are considered first. After the behavior of these networks is understood simulations of non-ideal systems are performed and deviations of simulation results from ideal network behavior are discussed.

Using the load model proposed in the previous section the processors can be considered to be operating in a compute-request-wait cycle. To increase processor utilization it is desirable to decrease to amount of time that a processor spends in the wait segment of the cycle. This time can be reduced to zero, thus raising $U_p$ to 1, if the delay incurred by remote service requests can be completely overlapped with local computation. The degree of overlap which can occur in a given architecture is determined by $O_{\text{max}}$. At this point it is convenient to divide the analysis into two cases; the first considers $O_{\text{max}} = 1$ in which no overlap occurs, and the second considers $O_{\text{max}} > 1$ in which the degree of overlap is influenced by other system parameters.
Analysis: $O_{\text{max}} = 1$

In this case no overlap can occur between local processing and remote processing. Consequently processor utilization must be less than unity if requests for remote service occur. Because a processor's requests are not pipelined the processor utilization is determined by the round-trip request delay. Under the logarithmic bus cost model discussed earlier, the minimum delay is independent of network bandwidth and is given by $D_{\text{min}} = 2n + R$. However, the degree to which the actual delay, $D$, exceeds $D_{\text{min}}$ is determined by the number of collisions in the system. Because for a constant load the number of collisions in the network varies inversely with the bandwidth of the network, $D$ should vary inversely with network bandwidth as well. Figure 3.17 shows an example of this behavior. Note that there is a region where $D$ is relatively constant and nearly equal to $D_{\text{min}}$. As $n'$ increases and several stages are replaced, the network becomes saturated and packets suffer larger delays. While $n'$ is small

![Figure 3.17: Delay vs. $n'$: $R = 2.0$](image-url)
the network is underutilized. A problem of interest is to determine at what point the delay begins to increase beyond $D_{\text{min}}$. This point determines how much network bandwidth is required to support a given system load and therefore allows the minimum cost network to be used. The following analysis assumes that there are no conflicts at the resources. This is not likely to be the case and resource conflicts act to increase packet delay. Therefore, simulations of networks show less network bandwidth required to maintain performance levels than the analytical results suggest.

If the system is to operate at its maximum performance level then the rate at which requests can be transferred to the resources must be greater than the rate at which requests are submitted to the network. The rate at which requests are submitted is determined by the delay through the network and by the amount of time a processor spends computing locally. Let $T_{\text{cycle}}$ be the time which elapses between successive request submissions to the network from a single processor.

$$T_{\text{cycle}} = \frac{1}{p} + D$$

$$\frac{1}{T_{\text{cycle}}} 2^{n'} < \frac{1}{n'}$$

(3.19)

$$n' 2^{n'} < \frac{1}{p} + D$$

As seen in Figure 3.17, in the region of constant delay $D$ is approximately equal to $D_{\text{min}}$. Since the desired network operates in this region, $D_{\text{min}}$ can be substituted for $D$. Equation (3.20) results by expanding the definition of $D_{\text{min}}$.

$$n' 2^{n'} < \frac{1}{p} + n + R + n$$

(3.20)

It is seen that for larger values of $R$ less network bandwidth is required to maintain performance levels. This agrees with results obtained from the probabilistic load model. As $R$ increases the performance of the interconnection network becomes less and less critical. Figure 3.18 shows the evaluation of equation (3.20) when a load parameterized by $O_{\text{max}} = 1$ and $p = 1.0$ is applied to an $N = 64$ system.

Equation (3.20) can be used to determine at what point $R$ becomes large enough so that a single bus provides sufficient network bandwidth. A bus network corresponds to the $n' = n$ case. By substituting this value into equation (3.20) and by noting that the $\frac{1}{p}$ term of equation (3.20) is minimized
Figure 3.18: \( n' \) vs. Minimum \( R (\log_2) \) to Avoid Saturation

when \( p = 1.0 \), equation (3.21) can be used to determine the minimum value of \( R \) which allows a bus to be used without degrading system performance.

\[
R > (N - 2) \log_2 N - 1 \tag{3.21}
\]

Equation (3.21) can also be rearranged to show the number of processors that can be supported by a single bus when \( R \) is fixed.

\[
(N - 2) \log_2 N < R + 1 \tag{3.22}
\]

Analysis: \( O_{\text{max}} > 1 \)

If \( O_{\text{max}} > 1 \) then overlap occurs between local and remote processing. Two questions considered in this section are:

(1) For a given load how much overlap can occur?
(2) What are the constraints on system parameters required to achieve the overlap?

A hybrid network used as an interconnection network with \( O_{\text{max}} > 1 \) acts as a pipeline for requests. The pipeline has an interstage delay, \( D_{\text{stage}} \), which is determined by either the resource service time or the bus transfer time, \( n' \).

\[
D_{\text{stage}} = \max (R, n') \tag{3.23}
\]

If the round-trip packet transfer time is given by \( D \) then the pipeline has \( \frac{D}{D_{\text{stage}}} \) stages; the pipeline is fully occupied when \( O_{\text{max}} = \frac{D}{D_{\text{stage}}} \). Because the pipeline cannot transfer packets faster than \( \frac{1}{D_{\text{stage}}} \), increasing \( O_{\text{max}} \) beyond this value provides no additional performance. Simulation results verifying this statement are shown in Figure 3.19. If \( O_{\text{max}} > 1 \) then the processor utilization of the system, \( U_p \), is determined by \( p \). To understand this let \( T_{\text{reply}} = \frac{D}{O_{\text{max}}} \). \( T_{\text{reply}} \) is the time between successive replies.

---

**Figure 3.19:** \( U_p \) vs. \( O_{\text{max}} \); \( n' = 2, R = 6.0 \)
returning to a given processor. If \( \frac{1}{p} \geq T_{\text{reply}} \) then the processor will be totally utilized. However, if \( \frac{1}{p} < T_{\text{reply}} \) then at times the processor will be idle and the utilization will fall off as indicated by equation (3.24).

\[
U_p = \frac{1/p}{T_{\text{reply}}}
\]

If \( O_{\text{max}} > 1 \) and \( U_p > 1 \) there are two situations. First, \( D_{\text{stage}} = R \). If this occurs then the network is not the bottleneck of the system and \( n' \) may be increased until either \( n' = R \) or \( n'2^{n'} = \frac{1}{p} \). Increasing \( n' \) in this manner will not increase system performance but it will decrease the cost of the interconnection network. In the second situation \( D_{\text{stage}} = n' \). Here the network is the bottleneck of the system and \( n' \) must be decreased until \( D_{\text{stage}} = R \).

To summarize - the interconnection network is viewed as a pipeline. It is desirable to keep delay through the pipeline small. Large queuing delays are prevented if the rate requests are submitted to the network, \( p \), is slower than the rate at which they can be transferred, \( \min \left( \frac{1}{R}, \frac{1}{n'2^{n'}} \right) \). Also, the throughput of the pipeline should be large. Since the throughput is determined by \( T_{\text{reply}} \), \( n' \) should not cause \( D_{\text{stage}} \) to exceed this value. The conclusion of this is that to minimize network cost without sacrificing system performance \( n' \) should be made as large as possible, subject to two constraints:

\[
\begin{align*}
n'2^{n'} & \leq \frac{1}{p} \\
n' & \leq \frac{D}{O_{\text{max}}}
\end{align*}
\]

Summary

Interconnection networks should not be considered in isolation. Arguments presented in Chapter 2 conclude that the effects of network reply time have a profound effect on network performance. A system model was developed which incorporates two features, \( k \) depth buffering, and non-zero reply times, which have not been reported in literature published to date. The model was shown to closely approximate behavior of a network simulator. The advantage of using the model to obtain
performance data is that its solution is computationally inexpensive relative to the cost of simulation. Explanations of the deviation of the model from performance measurements obtained by simulation were considered.

Next the model was generalized to predict the behavior of hybrid networks, delta networks which have had buses substituted for one or more of the final network stages. Good agreement between the model and simulations was observed if the bus transfer delay cost was assumed to be fixed in all networks. The model was less accurate in predicting the performance of the hybrid system when a logarithmic delay was used to model bus transfer cost. The logarithmic cost model is more realistic because arbitration costs are not constant with respect to the number of sources being arbitrated. Simulation studies were used to characterize hybrid networks so that a system model with increased complexity, and therefore accuracy, could be used. These results showed that multiple levels of network performance can be obtained by varying the number of stages removed from a delta network. Consideration of ideal hybrid systems showed that as $R$ grows, the interconnection network no longer is the system bottleneck and thus stages can be removed from the network without significantly affecting system performance. By removing stages two things happen. The speed of the interconnection network becomes matched to the reply time of the resources and system cost is reduced. Values of $R$ which represent these crossover points were determined. Simulation results also demonstrated that in non-ideal systems network bandwidth is severely degraded unless buffering is present both at the resources and at the switches within the network.

The final result of the chapter was to investigate system performance when the system load is modeled in a delay driven manner. Using this load model, processor operation is modeled by a three part cycle characterized by two parameters, $p$ and $O_{\text{max}}$. In the first part of the cycle the processor performs some local computation which does not require use of the interconnection network or any resources attached to the network. The period of this phase is modeled by a geometric distribution with parameter $p$. In the second phase, the processor issues a request for remote service to the network. The third phase of operation models the limited concurrency found in the load. If $O_{\text{max}}$ requests issued from the processor are outstanding, an outstanding request being a request for which a reply
from the resource has not been received, then the processor blocks until the number of outstanding requests falls below the $O_{\text{max}}$ level. At that time the processor begins a new cycle.

The advantage in using this model of the system load is that the parameters describing the load are decoupled from the performance and architecture of both the interconnection network and the resources. This decoupling allows a constant load to be presented to various network and resource architectures. In this manner it is possible to determine the network bandwidth required to support a particular problem.

The amount of bandwidth required to support a computation was seen to be dependent on both $p$ and $O_{\text{max}}$. It was shown that $p$ must be less than a critical value determined by the system bottleneck in order to prevent the network from saturating. This is important because when the network becomes saturated, poor performance results due to long queuing delays. The results also concluded that when the network is viewed as a pipeline, $O_{\text{max}}$ determines the degree to which the pipeline stages are occupied. If $O_{\text{max}} < \frac{D_{\text{stage}}}{D}$ then processor utilizations less than one can result.
CHAPTER 4

Resource Scheduling Interconnection Networks

As the use of distributed computing and parallel processing become increasingly common in computer architectures, a trend toward the use of auxiliary hardware modules is also evident. These functional units have their architectures tailored to perform specific tasks efficiently. Examples include FFT modules, high-speed floating point computation [20,4], image processing systems [19], graphics engines, and systolic arrays. These customized hardware modules can be viewed as resources which are shared among the set of general purpose processors in a multiprocessor system. Processors utilize the resources by making requests for service to the resources. Resources detect these requests and perform the necessary computations. Results are then returned to the appropriate destinations. If this mechanism is used in an intelligent manner then increased performance can result due to the specialized nature of the resource as well as from increased concurrency achieved by overlapping computation at the resources with independent computation performed by the processors.

In this type of architecture the interconnection network is used to create and maintain a communication channel between a processor requesting service and a resource available to provide that service. This type of network has been termed a resource scheduling interconnection network, or RSIN, by previous investigators [29]. A RSIN differs from conventional interconnection networks in a fundamental way. Because only a service is requested from the pool of resources, any resource which can provide that service may be utilized. Therefore there is no longer a unique destination requirement imposed on the request. The path that a request takes through the RSIN is not necessarily determined by an address tag associated with the request and in fact, routing may be performed using a completely stochastic algorithm.

Previous work by Jump et al. [16] and Wolf [18,30] among others, has shown that by allowing relatively complex operations to be performed by remote functional units significant performance gains can be achieved for some problems. The operating environment described in Chapter 2, namely
a network interconnecting processors with servers which have large service times, accurately describes an RSIN based architecture. One proposed architecture, the Rice Array Processor [16, 31], is based on an RSIN interconnection; resource service times are expected to be an order of magnitude longer than the interconnection network transfer delay.

Related Work

Jenevein et al. [32], while not considering scheduling networks, presented a discussion of hardware which can be used to support the allocation of resources to processors in multiprocessors based on banyan or tree interconnections. The basis of this work is the TRAC project at the University of Texas. The authors show how a comparator tree can be used to allocate processors and resources to a given task. Unfortunately, this mechanism is performed in a centralized manner and external to the system interconnection network. It is preferable that the network itself perform the allocation of resources and that this allocation be performed in a distributed fashion.

Wah and Hicks [29] have addressed this problem. They consider both crossbar and omega networks [33] and the the performance of distributed scheduling algorithms on each. Their performance evaluation considers two modes of operation. First they consider the case where the processor/resource connection must be maintained for the complete resource service time. Second, they consider the case where a request for service is sent to the resource and then the path is released. The performance evaluation consists of selecting a random set of processors and resources and finding the maximum number of resource allocations which can be made. A blocking probability is then calculated as:

\[
Pr_{\text{blocking}} = \frac{\# \text{ allocated processors}}{\# \text{ requesting processors}}
\]

The analysis consists only of this static, combinatorial type of investigation. No consideration is given to the dynamic behavior of the network.

Juang and Wah [34] have considered the problem of optimal scheduling on a circuit switched multistage network. The approach they take is to convert the routing problem into a flow problem. The network is transformed into a flow graph and various algorithms can then be used to determine the
maximum flow which can be pushed through the graph. Extensions to the homogeneous resource
model are presented to allow heterogeneous resources but no performance data is discussed for either
case.

Wah [35] addresses some of the performance questions with respect to RSINs. In this paper
three classes of networks are considered: the single bus, multiple shared buses or crossbars, and mul-
tistage networks. Network operation in all cases is considered to be circuit switched with each proces-
sor requiring only a single resource. Resource service time and network transmission time are both
assumed to be exponentially distributed. Wah first considers the performance of the single bus system.
He states that for the lightly loaded case the system behaves as an M/M/1 queuing system with the bus
acting as the bottleneck. For the heavily loaded case he claims the system is modeled by an M/M/1
queuing network with the resources forming the bottleneck. Investigation of intermediate cases is per-
formed by creating a Markov chain which is then solved to find the average queuing delay. Next, he
considers the operation of a RSIN consisting of multiple shared buses. The light load performance is
approximated by the analysis performed on the single shared bus case. Arguments are presented to
show that the Markov chain analysis can be used to approximate the heavy load case. Wah uses simu-
lation to investigate moderate load cases. Likewise, simulation is used to investigate the multistage
network system. In the final section of the paper, Wah states some guidelines which can be used to
determine which of the three network topologies should be used in a given architecture.

In this chapter the work outlined in the previous section will be extended. First, the model
developed in the previous chapter for address routed networks will be adapted to model RSIN
behavior. Performance predicted by the model will be compared to simulation results. Second, both
the model and simulations will be used to characterize the behavior of hybrid networks used as
scheduling networks.

Multistage RSIN Model

The fact that messages are no longer constrained to reach a particular destination forces some
changes to be made to the numerical model as developed thus far. The first change occurs in the
model of the resources. Because service can be provided by any resource the state transition probabilities developed in Chapter 3 no longer hold. Instead, the $S$ resource system which is on a common bus can be in one of $S+2$ states. The system state is given by the number of resources which are currently busy. State $S+1$ is entered when all resources are busy and an arrival occurs at the input of the system. This is the "blocked" condition which allows the RSIN model to fit into the framework of the model developed earlier. Using these state definitions combined with the geometric service interval of the resources, the equilibrium state probabilities can be constructed from equations (4.1-4.4) in a manner analogous to that described previously.

\[ Pr \left( S+1 \rightarrow S+1 \right) = (1-r)^S \]  \hspace{1cm} (4.1)

\[ Pr \left( S \rightarrow S+1 \right) = p(1-r)^S \]  \hspace{1cm} (4.2)

In equations (4.2-4.3) let $q$ and $q'$ represent the current state and the next state respectively with $q, q' \leq S$.

\[ Pr \left( S+1 \rightarrow q' \right) = \begin{pmatrix} S \\ q'-1 \end{pmatrix} (1-r)^{q'-1} r^{S-(q'-1)} \]  \hspace{1cm} (4.3)

\[ Pr \left( q \rightarrow q' \right) = (1-p) \begin{pmatrix} q \\ q'-1 \end{pmatrix} (1-r)^{q'-1} r^q + p \begin{pmatrix} q \\ q'-1 \end{pmatrix} (1-r)^{q'-1} r^{q-(q'-1)} \]  \hspace{1cm} (4.4)

The second change forced by the removal of the unique destination requirement lies in the model of the routing policy at the switches. No longer is there an address tag associated with each message which determines the necessary switch setting. This implies that some routing policy must be applied to messages arriving at a switch. It is desirable that this policy be implemented in a distributed manner for two reasons. First, if a centralized scheduler is used then it becomes a potential system bottleneck. Second, if the routing is done in a distributed manner then the system remains more modular and expandable. It is also desirable that the routing policy require as little global state information as possible since the communication of such information creates extra overhead in the system. It is also desirable to use as little additional hardware as possible so that network cost is minimized. In general, this routing problem is difficult to optimize but in a related problem, load sharing in a distri-
buted system, Eager et al. [36] have shown that simple policies for routing excess local loads to remote systems yield performance close to that of complex routing policies. This result, combined with the fact that networks investigated here are used in tightly-coupled systems where minimum overhead is essential suggests that a simple probabilistic policy could be used. The coin-flip routing policy, $CFR$, routes messages by randomly selecting an output port with an equal probability of choosing any particular port.

It is easily seen that the use of $CFR$ is equivalent to the use of address tags when the assumption of uniformly distributed destination addresses is made. The argument is as follows. At each switch there is an equal number of destinations reachable from each output. Since the probability that a message is routed to a particular destination is independent of the particular destination, it follows that there is an equal probability that the required destination is reached via the upper output or the lower output. Since $CFR$ is equivalent to the addressed routing used in Chapter 3, in this case the only required change to the model is the change in the resource state transition probabilities. Figure 4.1 shows the performance of the RSIN model using a $CFR$ policy. Quite clearly the performance is degraded for small values of $R$. Since no collisions occur at the resources the degradation must be attributable to collisions at the switches. Many of these collisions can be avoided by a more intelligent routing policy. $CFR$ does not take advantage of any available status information and therefore is far from an optimal policy. In fact, there is no reason any collisions should occur in the network. Since in a network composed of $c \times c$ switches there are at most $c$ messages waiting at the head of input queues and since there are also $c$ output links on which to route messages there is always an output available unless the buffers in the following stage are filled. A much better policy was proposed by Wah and Hicks [29]. They propose the use of feedback signals to allow a stage which has full buffers to communicate this fact to the preceding stage of the network. In this manner switches can avoid blocking due to full buffers as well as avoiding collisions within the switches. The disadvantage of this policy is two-fold. First, there is additional cost involved in constructing the network since more complex switches must be used to detect and propagate the control signals. Second, the network may operate at slower speeds due to timing requirements imposed on the system by the additional control.
signals. In the worst-case a buffer-full signal may need to propagate from the resources completely back through the network to the inputs.

As a compromise solution the optimal routing policy, OPT, is proposed. While it is seen that this policy is not truly optimal because it is not able to route messages away from full buffers, it is optimal in the sense that no collisions occur. If $R = 1.0$ then the OPT policy performs as well as the scheme proposed by Wah and Hicks without the added expense of more complex handshaking. OPT routing simply sends the message to the output which last successfully transmitted a message. This is different from round-robin scheduling in that successive messages may be directed to the same output. It performs better than round-robin scheduling since once an output blocks due to full buffers in the next stage subsequent messages are directed to another output until the blocked output clears. When round-robin scheduling is used the potential exists for routing a message to a full buffer while an alter-
native path would have lead to a non-full buffer.

OPT routing can be integrated into the framework of the existing numerical model quite easily. The only change required is to set the single switch blocking probability, $b$, to zero for all arrival rates. As was mentioned in a preceding paragraph, blocking still occurs due to full buffers; in hybrid networks blocking also occurs due to contention for the resource buses. Figure 4.2 show the performance of the model under the OPT routing policy. Consideration of Figure 4.1 and Figure 4.2 shows that while the choice of routing policies has a large effect on performance for small values of $R$, the effect is much less as $R$ increases. For $R \geq 5.0$ the effect is negligible. This means that in many RSIN based systems the choice of routing policies should be based on ease of implementation.

It is of interest to investigate the performance of hybrid networks which are operated as RSINs. Introductory discussion stated several examples of systems centered around this type of network.

---

![Figure 4.2: NTP vs. $p$; OPT Routing policy](image-url)
Several investigations have anticipated large service times; in these cases it may be possible to reduce system cost without reducing system performance by using hybrid networks in place of delta networks in the same manner as was proposed in the previous chapter.

Hybrid Network RSIN Performance

Figure 4.3 shows the $NTP$ of a system using $OPT$ routing with $R = 2.0$ for $\alpha' = 0, 1, \text{ and } 2$. As expected from results presented in Chapter 3 the performance levels of the $\alpha' = 0$ and the $\alpha' = 1$ systems are nearly equivalent. This is due to the fact that the resource bus does not become the system bottleneck until $\alpha' = 2$. One unexpected characteristic of Figure 4.3 is that performance increases when $\alpha'$ changes from 0 to 1. This is explained by the fact that the bus delay is modeled by a constant cost. Therefore when the network is shortened by a stage the blocking probability and the network delay both decrease. The result is a slight increase in $NTP$. From this point on simulation will be used.

Figure 4.3: $NTP$ vs. $p$
to obtain performance measures so that the logarithmic cost model of the bus can be used as well as the delay driven model of the load.

Delay Driven RSIN Performance

The preceding sections of this chapter have shown that when using a strictly probabilistic load model the behavior of RSIN networks is quite similar to the behavior of address routed networks. To determine if the similarity remains when the delay driven model is used, several experiments performed in Chapter 3 will be repeated using RSIN networks.

Again the investigation is divided into two cases. First the $O_{\max} = 1$ case is considered. Figure 4.4 shows processor utilization as a function of the reduction in network bandwidth. As long as the constraint on $n'$ given by equation (3.20) holds, $U_p$ remains high. However, when $n'$ is increased

![Figure 4.4: $U_p$ vs. $n'$: $O_{\max} = 1$](image)

Figure 4.4: $U_p$ vs. $n'$: $O_{\max} = 1$
beyond that point (in this example, beyond \( n' = 2 \)) performance falls as the bus saturates.

When the system load is characterized by \( O_{\text{max}} > 1 \) two constraints must be met. These were given by equations (3.25-3.26). First, the resource bus must not be saturated; second, the resource bus transfer delay must not be the system bottleneck. These constraints also hold for RSIN networks as shown by Figure 4.5 and Figure 4.6. Figure 4.5 shows that resource bus saturation remains a factor in system performance. \( U_p \) drops when \( n' \) reaches 2. This is predicted by equation (3.25); at this point \( n'2^{n'} = 8 \) while \( \frac{1}{p} = 5 \). Figure 4.6 demonstrates the need to keep the delay across the bus smaller than the service time at the resources. At \( n' = 2 \) the resource bus becomes the system bottleneck. Both of these constraints are intuitive; particularly in light of the analysis presented in Chapter 3.

---

**Figure 4.5:** \( U_p \) vs. \( n' \); \( p = 0.2 \)
Summary

The significant result of this chapter is the extension of the address routed model developed in Chapter 3 to resource scheduling networks. The model was then used to investigate one aspect of RSIN design, routing policies. Two routing policies were compared, CFR and OPT. The predicted performance level in systems with $R = 1.0$ was much higher for OPT than for CFR. However, as $R$ increased the difference between the two policies diminished until for $R = 5.0$ they were nearly equivalent. This indicates that the choice of routing policies is not critical in systems with moderate to large resource service times.

Next the behavior of hybrid networks used in an RSIN mode was considered. It was seen that the amount of bandwidth required to support a computation (based on the delay driven load model) was the same as for the address routed networks. This was not surprising since the address routed
analysis was based on ideal networks, networks which do not suffer collisions. If $OPT$ routing is used then there are no collisions in RSIN networks and therefore their behavior should follow the analysis of Chapter 3 even more closely than does the behavior of the addressed networks.
CHAPTER 5

Network Implementations under VLSI Constraints

The combined explosion of the use of VLSI technology to implement design solutions and of the demand for fast multiprocessor architectures has led to the need to understand problems encountered when employing VLSI techniques to design interconnection networks. These problems include the choice of network topology, network partitioning over multiple integrated circuit packages, and how to best utilize restricted amounts of package I/O bandwidth.

The previous chapters have shown that in a system which is network centered, systems in which the network is a central resource, often the bandwidth of a delta network cannot be fully utilized. Given this fact, it is clear the expense of a large crossbar network is not justified and consideration can be restricted to the bus and delta network interconnections. The chapter begins with a brief overview of previous work concerning some of the problems of using VLSI to build interconnection networks.

Previous Work and Chapter Overview

The foundation work in this area is a paper by Franklin [37]. This paper presents a comparison of the VLSI, single package implementations of a crossbar network and a multistage network. The results show that contrary to SSI implementations, the area complexity of both networks is approximately equal, $O(N^2)$ where $N$ is the number of input channels to the switch. Delay comparisons are also made but suffer from the fact that no blocking delays are considered. This biases Franklin's analysis in favor of the delta network. In actual operation the delay of the delta network will be higher than presented due to the fact that delta networks are blocking networks. Crossbar networks, which are non-blocking, do not suffer this increase in delay. Additionally, the network investigated by Franklin is unbuffered. The following investigation considers buffered networks.

Franklin et al. [38] extend the work presented in the paper discussed above. This work discusses the pin limit problem. As was briefly discussed in the introductory chapters of this
investigation, the factor which limits the size of single package network implementations is not chip area. Rather, the limiting factor is the number of pins which can be used to perform network I/O. The authors point out that a square network of size 32 which uses 16 bit wide data paths requires 1024 pins for data transfer. This is much larger than current conventional packaging techniques permit. Clearly, one solution is to use bit-slice techniques to distribute a network over several packages. The question of how to partition a network in an optimal manner is the primary result presented in [38]. Optimality is determined with respect to the product of chip count and average message delay. Both crossbar networks and multistage networks are investigated. These networks are constructed from smaller networks which are implemented in single packages using crossbar topologies. Again, the network is operated in an unbuffered mode. Other assumptions which are made include short messages and fast servers. These assumptions may not be valid when the pipelined operation of buffered networks is considered.

Szymanski [39] considers the problem introduced by Franklin's first paper [37]. He argues that the area complexity of banyan networks can be reduced through the use of switch-recursive banyan networks rather than the SW-banyan networks considered by Franklin. An SR-banyan network differs from a SW-banyan in that the switching points within the network are implemented by smaller banyan networks rather than by crossbar networks. The reduction in area complexity is achieved by reducing wiring complexity. Claims are made that area can be reduced by 25% over that of a SW-banyan. One flaw in this paper is that the limit on network integration is imposed by I/O constraints. Decreasing required chip area does nothing to modify this limitation. The paper does not address pin limit problems. However, the results presented by Szymanski are useful in routing delta network interconnections on the circuit board level.

A more comprehensive and detailed design study of VLSI networks is presented by Franklin and Dhar [10]. The authors consider in great detail the design of large multistage networks. Some of the results presented include:

1. Single package network size is limited by the number of available pins rather than by the size of the circuitry required. This is determined using a liberal estimate for λ, the VLSI line width, so
improvements in lithographic techniques will only compound the problem.

(2) Using an estimate of 240 pins per package and a data path width of 4, a $16 \times 16$ crossbar network can be implemented in a single package.

(3) The round-trip delay through their network is very large. The authors fail to point out that the network is in effect pipelined and that the rate at which results return from the network may be faster than the time quoted for the round-trip delay. This pipelining is of course subject to the characteristics of the load presented to the network. Round-trip operation is considered as is a fixed service time at the destination side of the interconnection network.

The analysis presented in this chapter uses the delay driven load model proposed in Chapter 3. Performance of a given interconnection scheme will be determined by the $NTP$ of the network. Network cost evaluations are based on the assumption that the switch modules represent the majority of the network cost. Unless otherwise noted, networks evaluated connect 64 processors with 64 resources.

The first section of this chapter investigates the performance of a delta network in which a message is broken into multiple packets for transmission through the network. The performance is compared to that of a bus which is capable of transmitting a message in a single cycle. This analysis will show that the bus network, despite its wide data path, chokes system performance. This occurs even at very low request rates.

Given that more interconnection bandwidth is required than a bus can deliver, how should a delta network be constructed? From previous authors' results it is clear that package pin limitations cause network partitioning to be a real concern. The second section considers one aspect of this problem.

The third section considers the problem of excess network bandwidth from a different vantage point. In Chapter 3 network bandwidth was matched to resource service time by removing stages of the network. In the third section, an attempt is made to match network operation to resource operation by reducing network data path width. This increases the number of cycles required to move a com-
plete message through the network.

The fourth section compares the two methods of matching system operating points. What are the advantages in cost and performance of each technique?

Narrow Delta Networks - Wide Buses

Franklin and Dhar state in their network design [10] that using $16 \times 16 \times 4$ switches, an entire circuit board is required to implement a $256 \times 256$ network with data paths 4 bits wide. This is a large amount of hardware which serves to increase system cost and decrease system reliability. It is therefore reasonable to ask under what circumstances a wide bus network might be more desirable than a relatively narrow delta network.

First it is necessary to consider the factors determining the performance of systems using these networks.

1. **Network size.** The cost of a bus interconnection grows as $O(N)$ but its performance decreases as $N$ grows, or at least remains constant. This means that as $N$ grows, bandwidth per processor falls linearly at best. On the other hand, the cost of a delta network increases as $O(N \log N)$ but the available network bandwidth per processor remains constant. Thus if the bus is to remain a viable interconnection as $N$ increases, the traffic on the network must decrease.

2. **Network speed parameters.** These are somewhat difficult to determine due to the difference in the way the networks operate. The network speeds represent the "implementation" of the networks and must be equivalent for the comparison of performance data to be valid. Delays incurred due to arbitration for resources, the bus and network links, are modeled as being proportional to the number of sources being arbitrated; for example, a bus with 4 sources to be arbitrated suffers twice the arbitration delay of a bus with only 2 sources being arbitrated. Note that a source is counted whether or not it is contending for the resource in the current cycle.

3. **System load.** If relatively few sources of traffic on the network are active at a given time then the parallelism present in the delta network will not be utilized. Conversely, if the load is high then the sequential nature of the bus will cause long waiting times at the sources.
Before presenting performance results it is necessary to discuss the model being used. The object of this section is to determine when a bus network which has a data path width of \( \omega_b \) performs better than a delta network which has a data path width of \( \omega_\delta \). A message is assumed to be \( \omega_b \) bits in length. The ratio of the widths, \( \omega_b \) is indicated by \( \omega \), where \( \omega \geq 1 \). Because the delta network has narrower paths transferring a message requires multiple packets. The delta network performs these transfers in a message switched mode. Message switching means that all packets of a common message travel together. The initial packet of each message acquires the required switch output and is routed to the following stage. All intermediate packets follow the path determined by the initial packet. The last packet in the message releases the switch output so that subsequent messages can be routed using that output. The performance of this mode of operation has been considered in detail by Walkup [13].

The operation of the resources is also affected by the multipacket message format. In this model a resource cannot begin providing service until all packets of a message have been received. Also, all packets of the reply message must be submitted to the reply network before a resource can begin to accept packets from the next request for service. This acts to increase the service time for resources interconnected by a delta network. If \( R_\delta \) represents the effective service time for a delta network in which \( \omega \) packets are required to send a message then the following expression holds:

\[
R_\delta = R + \omega + \omega
\]

Using this model the ideal throughputs of the two networks can be compared. The transfer rate of the bus can be limited by one of two factors. First, the system may be bus limited in which case the bus is capable of a single cycle every \( \frac{1}{\log_2 N} \) time units (logarithmic arbitration). This yields a throughput of \( \frac{1}{N \log_2 N} \). Second, the bus may be resource limited in which case the bus performs \( N \) transfers every \( R \) time units leading to a throughput of \( \frac{1}{R} \). The bottleneck in the delta network is the resource service time. These arguments lead to the following:
\[
NTP_b = \frac{1}{\max (R, N \log_2 N)} \\
NTP_\delta = \frac{N}{R_\delta} = \frac{N}{R + 2\omega}
\]

In most systems with moderate to large sized networks, \( R < N \log_2 N \). In this case the bus provides more bandwidth than the delta network when \( \omega > \frac{N^2 \log_2 N - R}{2} \). If \( N \) is moderately valued it is not feasible to build a bus with the required width.

The other situation when a bus may be sufficient occurs when the network load is very light. Using the delay driven load model and assuming that \( O_{\max} = 1 \) it can be determined how large \( p \) can be before the bus saturates. It was shown in Chapter 3 that when a network is operated in the saturated region unacceptably large message delays are introduced.

If \( \frac{1}{p} > N \log_2 N \) then the bus is not saturated and the processor utilization is maximized. In this situation a bus is an acceptable interconnection network. Unfortunately, many systems do not exhibit values of \( p \) which meet this criterion. For example, consider the gross analysis of a 64 processor system which uses a bus to fetch blocks to local cache from a shared memory. Assume that the bus is the width of a cache block, there are no writes to the memory from cache, and there is no memory latency. These conditions imply that the only bus traffic is data being read into the cache on misses. Even if the caches have hit ratios of 0.99, an extremely good cache, the bus will be saturated for a 32 processor system. In moderately sized multiprocessors, more bandwidth is required than a single bus can provide.

**Network Partitioning**

The previous section establishes the need for parallel interconnection networks. This section addresses one problem related to the implementation of delta networks. In previous work outlined in this chapter's introduction it has been determined that the number of pins on a package is the limiting factor in constructing delta networks using VLSI. Given this fact, is it better to have many channels per chip, each of which has a narrow data path, or is it better to have a few channels per chip, each of which has a wide data path? For example, if 128 pins are dedicated to data transfer, does a 64×64
network constructed from $2 \times 2 \times 32$ switches result in better performance than a network constructed from $64 \times 64 \times 1$ switches?

This section begins by discussing a detailed model of network delay which reflects the tradeoffs between switch size and the number of stages in a network. Next, the effects of switch size on data path width are reviewed. Finally, using this model and the delay driven model of the system load it is shown how to choose a switch size that maximizes processor utilization. These results are then compared with previous work in the area.

While Franklin et al. [38] have investigated network partitioning, their performance evaluation is rather cursory. They present an excellent analysis and development of VLSI circuit details but their performance analysis is restricted to a determination of the source to destination delay; they also restrict their results to unbuffered networks. In this section, the use of the delay driven load model yields a wider range of performance measurements than the heavy load model used by previous investigators.

In evaluating network partitioning performance differences are caused by trading switch size against the number of stages used to implement the network. Therefore, the critical modeling decisions revolve around the determination of switch delay and interstage delay, $t_{\text{switch}}$ and $t_{\text{prop}}$. Previously in this dissertation both delays had been lumped together. This was satisfactory since switch structure had not been an object of investigation.

Fortunately, the detailed analysis presented in the reports discussed above can be used to determine the delays. \textit{Intraswitch delay}, assuming an MCC crossbar switch [10], has been shown to be $O(c)$, $c$ being the number of input channels to the switch. This delay is due to the fact that all signals traverse $c$ switching points within the package.

The \textit{interstage delay} is more difficult to determine as it involves the capacity of a device on the switch package to drive an external signal trace found on the network’s circuit board. It was shown in [38] that propagation delay grows logarithmically with the signal path capacitance. Signal path capacitance in turn grows linearly with signal path length. Unfortunately, the path length is dependent upon the particular stage being considered as well as the particular stage input/output pair. One approximation is to model the signal path length as being proportional to the number of switches in a
stage. This leads to a model of interstage delay which grows logarithmically in the size of the network and decreases linearly with switch size increase.

The delay through a network is modeled as the sum of two components, the intraswitch delay, $t_{\text{switch}}$, a delay equal to the number of channels into the switch modules, and the interstage delay, $t_{\text{prop}}$, a delay representing signal propagation between stages - equal to the base two logarithm of the number of switches in a stages. Using this delay model and an assumption of zero delay cost to enter and exit the network, Figure 5.1 shows the minimum delay incurred by a packet while it traverses the network in a single direction for several network partitions.

The use of the delay driven load model implies that in many cases the network is operating in lightly loaded conditions. In this situation the delay of the network has a strong effect on the performance of the system. In particular, if the network is viewed as a pipeline and the stages of the pipeline are occupied relatively infrequently then the utilization of the processors is inversely proportional to network delay. The conclusion reached from this argument is that the choice of network partitioning

<table>
<thead>
<tr>
<th>Network Size</th>
<th>Switch Size</th>
<th>Stages</th>
<th>$t_{\text{switch}}$</th>
<th>$t_{\text{prop}}$</th>
<th>$D_{\text{min}} = \text{stages} \cdot t_{\text{switch}} + (\text{stages} - 1) \cdot t_{\text{prop}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>4·2 + 3·3 = 17</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2·4 + 1·2 = 10</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>1</td>
<td>16</td>
<td>0</td>
<td>1·16 + 0·0 = 16</td>
</tr>
<tr>
<td>64</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>6·2 + 5·5 = 37</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3·4 + 2·4 = 20</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>2·8 + 1·3 = 19</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>1</td>
<td>64</td>
<td>0</td>
<td>1·64 + 0·0 = 64</td>
</tr>
<tr>
<td>256</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>7</td>
<td>8·2 + 7·7 = 65</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>4·4 + 3·6 = 34</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>2</td>
<td>16</td>
<td>4</td>
<td>2·16 + 1·4 = 36</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>1</td>
<td>256</td>
<td>0</td>
<td>1·256 + 0·0 = 256</td>
</tr>
<tr>
<td>4096</td>
<td>2</td>
<td>12</td>
<td>2</td>
<td>11</td>
<td>12·2 + 11·11 = 145</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>10</td>
<td>6·4 + 5·10 = 74</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>4</td>
<td>8</td>
<td>9</td>
<td>4·8 + 3·9 = 59</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>3</td>
<td>16</td>
<td>8</td>
<td>3·16 + 2·8 = 48</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>2</td>
<td>64</td>
<td>6</td>
<td>2·64 + 1·6 = 134</td>
</tr>
<tr>
<td></td>
<td>4096</td>
<td>1</td>
<td>4096</td>
<td>0</td>
<td>1·4096 + 0·0 = 4096</td>
</tr>
</tbody>
</table>

Figure 5.1: Partitions for Several Network Sizes
should be based on minimizing network delay. However, an algorithm to directly determine which switch size leads to the minimum network delay has not been found [38]. A useful heuristic is to select the switch size such that $t_{\text{prop}} = t_{\text{switch}}$.

Choosing the minimum delay network is optimal for $O_{\text{max}} = 1$ loads as was shown in Chapter 3. For $O_{\text{max}} > 1$ loads the situation is more complex. The added complexity arises from pipeline effects. No longer is it sufficient only to consider network delay, when the network is used to pipeline packets two additional considerations arise. First, the throughput of the network is limited by the slowest stage in the pipeline. This implies that $t_{\text{prop}}$ and $t_{\text{switch}}$ should be matched. Also, the value of $R$ must be considered. Decreasing either of the network delay parameters below $R$ will result in little improvement in system performance. The second consideration is the length of the pipeline. Networks with fewer stages are in effect shorter pipelines. Shorter pipelines cannot overlap as many operations as longer pipelines; therefore the throughput of a short pipeline is lower than that of a longer one. It is difficult to design an analytical model which captures the interdependence of these effects. Because of this, performance comparisons in this section are made using simulation results.

The first problem considered is to determine how traffic intensity affects the relative performance of networks constructed from different size switches. From the previous discussion and results of Chapter 3 it is hypothesized that at light loads network performance is determined by network delay. For a $64 \times 64$ network this means that networks of $8 \times 8$ and $4 \times 4$ switches should perform nearly equally and that both should perform better than a network of $2 \times 2$ switches. As the load increases the pipelining ability of the network becomes more critical and networks which match $t_{\text{switch}}$ and $t_{\text{prop}}$ should show the best performance. In the $64 \times 64$ network case this means that the $4 \times 4$ switches should outperform the other partitions. Figure 5.2 shows the $NTP$ of three $64 \times 64$ networks as a function of $O_{\text{max}}$. Increases in $O_{\text{max}}$ reflect increased traffic in the network. The experimental results support the arguments. An additional observation can be made from Figure 5.2. The value of $O_{\text{max}}$ which causes network saturation is different for all partitions. This can be attributed to the different number of stages in each network. A larger number of stages creates a longer pipeline which can take advantage of increased parallelism as reflected by larger values of $O_{\text{max}}$. 
Cheemalavagu and Malek [40] have investigated the performance of rectangular banyan networks constructed from various switch sizes. Delta networks are a type of banyan network. It is difficult to interpret the results presented in this paper because the description of network operation is non-existent. In particular, their choice of interstage delay models is not presented. Despite this, the authors conclude that $4 \times 4$ switches are the best choice of switch size. However, the comparison presented is biased because the cost of the networks is not considered. The cost of a $64 \times 64$ network constructed from $8 \times 8$ switches is likely to be less than the cost of a $4 \times 4$ partitioning because fewer switches are required.

When VLSI is used to implement networks the cost of the network can be modeled as being proportional to the number of switches in the network. If the number of channels on each switch is $c$ then the number of switches required to form an $N \times N$ network is given by equation (5.1).
\[ \text{switches} = \frac{N}{c} \log_2 N \]  
(5.1)

Clearly the cost of building an \( N \times N \) network falls as the size of the switches increases. If this were the only effect of larger switch size then it would be optimal to build a single integrated crossbar switch. Unfortunately, the pin limit problem must be considered. Since there are a constant number of pins available for data transmission in a given type of packaging, the product of switch size and switch data path width must be constant. If the switch size doubles, the data path width is cut in half. Normalizing data path widths to the width of the paths on a \( 2 \times 2 \) switch so that data paths are now described as multiples of the width of paths on the \( 2 \times 2 \) switch, the path width on a \( c \times c \) switch is given by equation (5.2).

\[ \text{data path width} = \frac{2^c}{c} \]  
(5.2)

In order to provide an accurate evaluation of network partitions the networks being compared must have equivalent costs. If the cost of the networks to be evaluated is chosen to be the cost of a network implemented using \( 2 \times 2 \) switches then in networks employing larger switches several switches can be used in parallel to increase data path widths. Franklin and Dhar [10] have considered some of the problems encountered when using bit-slice techniques to widen data paths.

To determine how wide the data paths are in a network constructed of \( c \times c \) switches which is equal in cost to the network constructed of \( 2 \times 2 \) switches let \( k_c \) be the number of \( c \times c \) switches which are operated in parallel to form a wider data path.

\[ \frac{N}{2} \log_2 N = k_c \frac{N}{c} \log_2 N \]

\[ = k_c \frac{N}{c} \log_2 N \log_2 c \]  
(5.3)

\[ k_c = \frac{c}{2} \log_2 c \]  
(5.4)

Since each switch has a width of \( \frac{2}{c} \) the data path width in the network is \( \log_2 c \). This result indicates that a message passed through a network built from \( 2 \times 2 \) switches requires 3 times as many packets as an equal length message sent through a network of \( 8 \times 8 \) switches. Kruskal and Snir [21] have considered the effects of varying data path widths. Their performance analysis is restricted to determining
an approximation to the transit time for messages in networks which differ in switch size. In the next section results are presented which compare throughputs of these networks under the delay driven load model.

One of the deficiencies of Kruskal and Snir's analysis is that it fails to consider the increased congestion in networks using smaller switches. This increase in congestion is caused by the fact that if a fixed number of messages are circulating in two networks, the network which has the smaller switches, and therefore the narrower data paths, has more packets circulating in it than does the network with the larger switches. Additionally, their analysis is analytical and an assumption of infinite buffering is made to achieve a tractable result.

Consideration of the above arguments leads to a variation of the previous question. Rather than simply considering the performance of networks with equal data path widths but different partitions, now different partitions will be considered in the context of constant cost networks. It is anticipated that the $8 \times 8$ network will show much better relative performance due to its increased data path width. Similarly, the poor relative performance of the $2 \times 2$ partition will be further degraded due to increased traffic intensities. Figure 5.3 compares the performance of the three network partitions considered in Figure 5.2. Messages in the $8 \times 8$ network are two packets long, messages in the $4 \times 4$ network consist of 3 packets, and messages in the $2 \times 2$ network are composed of 6 packets. All networks are operated in a message switched mode. Message switching in buffered networks is analogous to circuit switching. The first packet of a message acquires a path from stage $i$ to stage $i+1$. That path is reserved and can only be used by later packets of the same message. When the final packet of the message finishes using the path, it is released to another message. In this manner the message acquires and then releases the set of links required to move through the network. The resources are required to receive complete messages before beginning to provide the requested service.

In Figure 5.3 it is seen that the $8 \times 8$ configuration performs the best although it is nearly equivalent to the performance of the $4 \times 4$ network. The performance of the $2 \times 2$ network is decidedly inferior to the others; this conclusion is in agreement with other results [21, 40]. The fact that $8 \times 8$ switches perform as well as $4 \times 4$ switches is in contradiction with previous work. This is caused
by the fact that relatively small buffers were used in the simulation \((k = 2)\). If larger buffers are used then simulation results are similar to those presented in [21]. Figure 5.4 shows the throughput of networks with 4 buffers at each switch. The dependence on buffer depth arises from two sources. First, in the \(4 \times 4\) network there are three packets per message. Increasing buffer depth from two to four allowed an entire message to be buffered at a switch. Since the \(8 \times 8\) network sends a message in two packets, the depth 2 buffers were sufficient to queue an entire message. The poor performance of \(2 \times 2\) switches cannot be attributed to buffer length. In experiments done with length 6 buffers, the \(2 \times 2\) switch network performance improved but was still significantly lower than the performance of the other networks.

The second dependency is caused by the pipelined operation of the network. As the buffer depth increases, the network approaches its maximum throughput. It was seen in Chapter 3 that the pipeline rate is limited by the longest delay in the system. In the \(8 \times 8\) network this delay is \(t_{\text{switch}}\)
which is considerably larger than either $t_{\text{switch}}$ or $t_{\text{prop}}$ for the $4 \times 4$ network.

These results give new insight into the question of network partitioning. Based on two observations, namely that intraswitch delay varies directly with the size of the switch and that interswitch delay varies directly with the number of switches in a network stage, the solution to the problem of network partitioning becomes straightforward. The network is viewed as a pipeline in which $O_{\text{max}}$ determines the number of stages in the pipeline. If $O_{\text{max}} = 1$ then system performance is maximized by minimizing round-trip delay. As $O_{\text{max}}$ increases, the degree of overlap in the pipeline also increases; in this case performance is maximized by matching the delays of the switches and the links. Using this framework and Figure 5.1, it is predicted that for the 4096 network the optimal switch size is $8 \times 8$ for heavy loads and $16 \times 16$ for light loads. Unfortunately it is not currently possible to simulate such a large network on available computer systems.
Narrow Path Bandwidth Reduction

In Chapter 3 it was shown that network bandwidth is not fully utilized when service times are greater than the interstage transfer time. Hybrid networks with lower bandwidths were used to replace delta networks so that interconnection network utilization was increased. Because the hybrid networks are also less expensive than delta networks system cost was reduced without causing system performance to fall.

Consideration of the model of networks implemented with VLSI which was presented in the previous section suggests another method of reducing system interconnection cost. Rather than shortening the network as was done with hybrid networks the network can be narrowed. Suppose a network is implemented using $8 \times 8$ switches. It was shown that $\frac{8}{2 \log_2 8}$, or 12, switches can be used in parallel to increase data path width without exceeding the cost of an equal size network constructed of $2 \times 2$ switches. If fewer than 12 switches are operated in parallel then the resulting network will have lower bandwidth but the cost of the network will fall as well. If the lost bandwidth was unutilized to begin with then there can be a reduction in system cost without a corresponding loss of system performance.

Using networks constructed of $4 \times 4$ switches experimental evidence shown in Figure 5.5 shows that processor utilization decreases as the data paths are narrowed. This can be partially attributed to the small resource service time. Because $R$ is small delays incurred by servers waiting for the arrival of all packets in a multipacket message can increase the time required to process a request by a substantial percentage of the normal service time. As the service time increases these delays become less significant and the performance degradation decreases. This is verified by Figure 5.6 which shows processor utilization for a system which has $R = 10.0$.

Two additional observations can be made. First, the degree of performance degradation suffered varies directly with the $O_{\text{max}}$ parameter of the load. This is expected and is explained by noting that for larger values of $O_{\text{max}}$ the network is more congested. By decreasing data path widths additional congestion is caused which in turn causes poorer performance. When $O_{\text{max}}$ is relatively small the net-
Figure 5.5: $U_p$ vs. Message Length; $R = 1.0$

$O_{\text{max}} = 1, 2, 3, 4, \text{ and } 5$
work is largely empty. The packets introduced into the network by narrowing the data paths do not significantly increase the level of congestion in the network. Second, the fact that system performance drops relatively sharply when the number of packets per message is increased beyond 2 suggests that perhaps an increase in buffering would smooth this behavior. Simulation of the $R = 10.0$ system when the buffer depth is changed from $k = 2$, as in Figure 5.6, to $k = 4$ show that smoothing does occur. It is important to note that the use of these extra buffers allows data path width, and thus network cost, to be reduced without affecting system performance. The additional buffers increase system performance for heavy loads; the results for lightly loaded networks show no change. This is explained by the fact that the reason performance suffers under light loads is different from the reason performance declines under heavy loads. Heavily loaded networks show decreased performance because collisions cause the network throughput to fall. Dias and Jump [14] have shown that buffering can increase network throughput in these cases.
Degradation in lightly loaded networks is not caused by packet collisions. In Chapter 3 it was shown that for light loads the critical factor in determining system performance is network delay. Since the resources are required to wait until all packets of a message are received before beginning service the delay incurred by waiting for additional packets decreases the throughput of the resources. Also, the resource must send more packets in the reply message when the data paths are narrow. Buffering does not decrease the delay in either situation and therefore has little effect on system performance.

Hybrid and Narrow Network Comparison

Having presented and discussed two methods for reducing system cost without reducing system performance it is useful to consider which technique is better. The hybrid network pared system cost by reducing the length of the network while the narrow delta network cut costs by reducing the width of the network. In this section cost will be measured by the number of switches in the network. It is assumed that VLSI technology is used to implement the switches and thus data path widths are subject to the constraints discussed previously in this chapter. Delay through the network will be modeled as a unit delay per stage. The delay cost for the bus in the hybrid network will be logarithmic; \(4 \times 4\) switches will be used in all networks with 4 switches being operated in parallel at each switching point in a non-reduced network. In the previous section this network was shown to exhibit the best performance for \(64 \times 64\) systems.

Before examining experimental results consider the effects of each method of bandwidth reduction. First consider narrowing the networks. In this case performance is degraded by the fact that multiple packets must be transmitted for each message. If the width of the data paths is reduced by a factor of \(m\) then the maximum number of packets in the system increases by \(m\) as well. The increased number of packets increases network congestion. Additionally, as was discussed in the comparison of wide bus performance with narrow delta network performance, \(R\) is effectively increased due to the fact that a resource is delayed while dequeuing multipacket messages from its input buffers before the service interval can begin.
If the network is shortened rather than narrowed performance may be degraded for other reasons. The first cause of degradation is bus saturation. As was discussed in Chapter 3 this problem occurs at relatively low loads. Second, using multiple resources on a common bus leads to resource underutilization due to bus blockages. This problem was seen in Chapter 3 where it was demonstrated that the use of buffers at the resource inputs can reduce performance degradation due to this effect.

The preceding discussion indicates that for moderate to heavy loads narrowing the network is the better approach to reducing network bandwidth. Figure 5.7 shows the cost/performance tradeoffs for both types of networks in this environment; as expected, less performance degradation occurs when the network is narrowed. The relatively high load on the network causes bandwidth reduction to have significant effects on system performance.

On the other hand, in lightly loaded systems the better method to reduce network cost is to use hybrid networks. From Figure 5.8 it can be seen that no system performance is lost when the number

![Figure 5.7: \( U_p \) vs. \( O_{max} \); \( p = 0.1 \)]
of stages is reduced from 3 to 1. Reducing the data path width causes performance to be degraded in the $O_{\text{max}} = 1$ case, albeit only by 5-10%.

A question which arises from this discussion is: for what range of $p$ is it best to use the hybrid networks? The answer to this question is shown in Figure 5.9 which shows the performance of the networks as a function of $p$. Solid lines represent hybrid network performance for $n' = 0, 1, 2$; dashed lines represent performance for narrow networks with $m = 0, 1, 2, 3$. Results from Chapter 3 explain the performance curves very nicely. The hybrid solution performs better in environments where the bus is not saturated. For $n' = 2$ this occurs at $p = \frac{1}{n' 4^{n'}} = \frac{1}{32} = 0.031$. If $p$ is greater than the saturation point then the narrow networks perform better because of large queuing delays in the hybrid network. If $p$ is less than the saturation point then hybrid networks have an advantage due to the fact that only a single packet must be dequeued at the resources and thus the effective resource service time is lower. The results shown are for service times of $R = 10.0$; for shorter service times the performance advantage of hybrid networks will be more pronounced due to the fact that the relative increase in effective resource service time of the narrow networks will be larger.

Figure 5.8: $U_p$ vs. $O_{\text{max}}; p = 0.01$
Summary

This chapter considered some of the effects of using VLSI technology to implement delta networks. In particular, the constraint of insufficient I/O bandwidth imposed by packaging limitations was investigated.

First, consideration was given to the fact that a wide bus might be able to supply as much interconnection bandwidth as a delta network with narrower data paths. It was shown that for moderately sized multiprocessors, the bus would be prohibitively wide.

Second, given the limited I/O bandwidth of VLSI packages, an investigation into the problem of partitioning a large network into smaller subnetworks was performed. Using a delay model based on switch size and data path trace length, combined with results from Chapter 3, it was shown that the best system performance is achieved for $O_{\text{max}} = 1$ loads when total network delay is minimized, and for $O_{\text{max}} > 1$ loads when switch delay and propagation delay are matched. It was also shown that...
reduced bandwidth networks could be formed by narrowing the width of network data paths.

The final problem investigated was to determine which of the two bandwidth reduction techniques, hybrid networks or narrow data paths, showed better performance. Simulation results showed that the answer to the question depended on the load applied to the system. Under loads which did not cause the hybrid network bus to saturate, the hybrid network was the better solution. However, as \( p \) increased beyond the saturation point, narrowing network data paths showed better performance.
CHAPTER 6

Conclusions

The goal of this dissertation was to determine how much interconnection network bandwidth is required to support a given computation in a particular system. If this determination can be made then system cost can be minimized without sacrificing system performance.

Previous investigations have shown that buffered delta networks can provide approximately the same network bandwidth as a crossbar for lower costs. For this reason the investigation presented here began by considering the behavior of delta networks.

The first step in understanding delta network behavior was to develop a numerical model which computed the throughput for various networks as a function of the arrival rate at network inputs. The model incorporated two features previously modeled only through simulation, finite depth buffering at switch inputs and service times at network destinations which may be greater than the interstage transfer delay of the network. Both features represent significant extensions of previous buffered delta network models. The finite depth buffering is important because the cost/performance tradeoffs of buffer depths must be understood when developing actual systems. The service time generalization is important because in future systems delta networks are likely to be used more generally than as simple processor/memory interconnections. Solution of the model involved the use of aggregation techniques where the throughput of the network was found by recursively solving the model of a single stage. The pitfall of several previous investigations, interstage state dependencies, was avoided by adjusting the state probabilities in each stage to account for the influences of the other stages. Simulation results indicated that the model captured the effects of both varied buffer depths and varied service times.

Performance results obtained from the solution of the model indicated two things. First, in systems which had resource service times greater than 1 (normalized to the interstage transfer delay), delta network bandwidth was not fully utilized. Second, the amount of bandwidth which was used was dependent upon the resource service time. These two observations indicated that:
(1) A lower bandwidth, and therefore less expensive, network could be used in place of the delta network in order to reduce system cost. This reduction in cost is achieved without a loss in system performance.

(2) A family of such networks is required so that the interconnection network can be matched to the resource service time.

Hybrid networks, formed by replacing stages in the delta network with buses, fulfills both needs. By replacing switch stages with buses, the bandwidth and cost of the network are both reduced; by varying the number of stages replaced, a family of networks with cost/performance tradeoffs is obtained.

The next step in determining how much bandwidth is required to support a given computation is to parameterize the system load. The conventional model of geometric interarrival times is insufficient for two reasons:

(1) The model does not reflect the limited degree of concurrency in most problems, i.e. a processor can only submit some number of requests to the network before the processor must wait for a reply to be returned.

(2) The performance of the interconnection network and of the resources affects the rate at which subsequent requests are submitted to the network; if the delay from request to reply is small then requests will be submitted more frequently than if the delay is large.

To more accurately reflect this behavior, a delay driven model of system load was proposed. In this model each processor performs some local computation, characterized by $p$, then submits a request for remote service to the network and then waits for the reply to the request. A generalization was made to allow $O_{\text{max}}$ requests to be outstanding at a given time.

Using the delay driven model for the load, rules were developed to determine how much network bandwidth was required to allow a calculation parameterized by $(p, O_{\text{max}})$ to be solved at its maximum rate.
The investigation then turned to considering the minimum bandwidth problem for networks used in a resource scheduling mode of operation. RSINs differ from delta networks in that a request can be routed to any available resource. The single destination requirement no longer holds. The model developed for delta networks was extended to model two RSIN routing policies, CFR and OPT. The model showed that RSIN performance under CFR is equivalent to address routing when the assumption of uniformly distributed destination addresses is used. OPT routing allowed higher performance to be obtained because there are no collisions, all blocking is caused by full buffers. When hybrid networks are used as RSINs, the resulting performance levels are slightly higher than for address routed hybrid networks. This is due to the absence of collisions at the resources. If OPT routing is used then system performance is nearly equal to ideal levels. Generally, it seems as though RSIN performance is easily understood by considering the behavior of address routed networks.

The final aspect of network bandwidth considered in this dissertation was that of actually implementing these networks using VLSI technology. Previous investigations have shown that moderate to large sized networks must be partitioned over multiple chips due to pin limitations in available packaging technologies. The problem of how to best partition a network was considered and it was demonstrated that when $O_{\text{max}} = 1$ the partition which minimized round-trip network delay gave the best system performance; when $O_{\text{max}} > 1$, performance was maximized when $t_{\text{select}}$ and $t_{\text{pass}}$ were matched. These partitions led to the use of switches larger than $2 \times 2$. In these cases, several switches must be operated in parallel to maintain a constant data path width. This leads to another technique to reduce network bandwidth and cost. Data paths can be narrowed by operating fewer switches in parallel. This degrades performance by requiring the use of multipacket messages. Finally, a comparison between the two methods of bandwidth reduction was presented. Results showed that the performance of the hybrid networks is superior under conditions which do not saturate the resource bus. If the load on the network is high enough to send the resource bus into saturation, then narrowing the data paths leads to better cost/performance tradeoffs.
Extensions and Elaborations

There are several directions that continuations of this work can follow.

Network model:

Continued work with the numerical model of the hybrid network might address some of these questions.

(1) The building block for the network model was chosen to be a 2 × 2 switch. Although not much time has been spent pursuing the question, a generalization of the model to allow the construction of a network from other sized switches may give some additional insight into questions raised in Chapter 5. The difficulty in this is in representing behavior of switches in which more than two input channels request the same output channel.

(2) The worst assumption in the model is probably the assumption that a message consists of a single packet. This assumption was removed later in the dissertation when simulation was used to generate performance comparisons. It would be desirable to incorporate multipacket messages into the numerical model.

(3) Ever since Pfister and Norton [41] presented their work on the "hot-spot" phenomenon it has been a subject of some discussion. It appears that incorporating non-uniform destination addressing into the model is non-trivial. Kumar and Pfister have done some analytical modeling of hot-spots [42] but most of the other results have been found by simulation.

(4) Another limitation of the model is an inability to model buffering at the resources. It seems as though this can be added to the existing framework although the number of states in the model of the resource system will almost surely increase sharply.

(5) A very difficult problem, and yet one which is likely to be critical to more detailed numerical modeling of networks is the modeling of deterministic times. These occur in at least two places. First, the assumption of geometric service times at the resources is poor, particularly if the resources are representing memory modules or other very deterministic objects. The second source of deterministic time is found in representing the bus arbitration and transfer costs. It
was seen that to get the model to approximate simulation results with any accuracy at all, it was necessary to assume a unit time cost for bus activity. This assumption is probably not as realistic as desired.

Delay driven models:

Another possible area of investigation is the refinement of the delay driven model of the load. There are two clear problems here.

(6) If an architecture is to be optimized for a set of applications the characteristics of those problems must be discovered. In particular, what are reasonable values of $O_{\text{max}}$ and $p$ for real problems. It would be even better if there were an automatic method of determining these values given an algorithm.

(7) The second problem relates back to the network model. Since a delay driven load seems to represent system behavior better than a strictly stochastic one, a better model of network behavior can be realized if it incorporates the delay driven feature. This involves transforming the existing open network model into a closed model.

Optical technologies:

Implicit in the results obtained in Chapter 5 is the behavior of electronic circuits and electrical signals. Although the technology is developing at this time, optical techniques may have a profound effect on the way intermodule communication is performed in future systems. Are the same problems encountered in optically based networks that are found in electrical networks? Since optical fibers can be frequency multiplexed, is the I/O problem eliminated?
References


