PERFORMANCE ANALYSIS OF
PARTITIONED MULTISTAGE CUBE NETWORK AND
ADAPTIVE ROUTED SINGLE-STAGE CUBE NETWORK

by

Jahng S. Park

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

APPROVED:

[Signatures]

Nathaniel J. Davis IV, Chairman

F. Gail Gray

Scott F. Midkiff

February, 1994
Blacksburg, Virginia
PERFORMANCE ANALYSIS OF
PARTITIONED MULTISTAGE CUBE NETWORK AND
ADAPTIVE ROUTED SINGLE-STAGE CUBE NETWORK

by

Jahng S. Park

Nathaniel J. Davis IV, Committee Chairman
Electrical Engineering

(ABSTRACT)

Parallel processing has emerged to meet today's high demand of computationally intensive applications. In a parallel processing machine, the interconnection network is a major factor affecting the machine's overall performance in terms of speed and versatility. This research presents an analysis of the partitioned multistage cube network (MSC) and the adaptive routed single-stage cube network (Cube). The parallel processing system with a partitioned MSC provides computing power for sub-users more conveniently and performs faster than the same system with randomly subdivided processing elements. The performances of input partitioned MSC and output partitioned MSC are analyzed by simulation. These figures are then compared to the performance figures of the MSC with randomly grouped processing elements. A Cube with an adaptive routing algorithm provides a faster and more fault-tolerant information transmission medium for a parallel processing system compared to a Cube with a static routing algorithm. The simulated performance results of the adaptive routed Cube and the static routed Cube are compared. The analysis is done assuming packet switched scheme with a concentration on network throughput. A description of the investigation, assumptions and factors used for the study, and the results of the simulation analysis are included.
Acknowledgments

As I finish several months of research for this thesis, I would like to thank several people who have helped me through months of frustration and elation. First, I would like to express my thanks to my advisor, Dr. Nathaniel J. Davis IV for his patience and guidance. He helped me with his suggestions to overcome the difficult times during my research. He also endured the reading and numerous iterations of writings needed to obtain the final product. I also would wish to thank my committee members, Dr. F. Gail Gray and Dr. Scott F. Midkiff, for their time and cooperation in this investigation.

Special thanks are in order to Vish Ramachandran and Rick Raines, Captain, USAF. Vish helped me to get acquainted with SLAM and gave me invaluable assistance on system setup and simulations. Rick was always there to answer questions I have about the simulation models which I developed based on the models of his thesis research. A word of thanks is due to Kimy Loka who helped me in obtaining numerous simulation runs and analyzing the simulation data.

I wish to thank my best friend Won Jae Lee in Seoul. His letters of encouragement helped me overcome the emotionally difficult times I had during the research. I like to thank my good friends, Jae Woong Hwang, Lori Crawford, Thu-Anh Nguyen, Jae Ho Hwang, Il Hwan Yang, Hee Soo Ko, Jae Hwan Kim, Min Sook Kim. Irene Kim, and Michell S. Kwon, for their words of support. I wish to thank Dr. Cho, Dr. Ha and their families for their support and guidance.

Finally, I wish to express my gratitude and love to my parents, Mr. and Mrs. Jhungseh Park, and my dear sisters, Hye Sohng and Hye Jun. Without their love and support, I would not be here completing this thesis. To show my appreciation for their support, I dedicate this thesis to my family.

Jahng S. Park
Table of Contents

Abstract ........................................................................................................................................... ii
Acknowledgments .............................................................................................................................. iii
Table of Contents ............................................................................................................................. iv
List of Figures ..................................................................................................................................... vii
List of Tables ...................................................................................................................................... ix
1. Introduction .................................................................................................................................... 1
   1.1 Background ................................................................................................................................ 1
   1.2 Research Goals ............................................................................................................................ 2
   1.3 Summary ...................................................................................................................................... 4
2. Introduction to Parallel Processing Systems .................................................................................... 5
   2.1 Introduction .................................................................................................................................... 5
   2.2 Parallel Processing Systems Classification Methodologies ............................................................. 5
      2.2.1 Flynn’s Taxonomy .................................................................................................................. 5
      2.2.2 Feng’s Taxonomy .................................................................................................................. 6
      2.2.3 Händler’s Taxonomy ............................................................................................................. 8
      2.2.4 Skillicorn’s Taxonomy ......................................................................................................... 8
   2.3 Overview of Interconnection Networks ......................................................................................... 11
      2.3.1 Traditional Network Designs ............................................................................................... 11
      2.3.2 Network Design Decisions .................................................................................................. 12
      2.3.3 Machine Architecture (Network’s Viewpoint) ................................................................... 14
      2.3.4 Examples of Interconnection Networks .............................................................................. 14
2.3.4.1 The Single-Stage Cube Network .............................................. 17
2.3.4.2 The Multistage Cube Network .................................................. 17
2.3.4.3 The Augmented Shuffle Exchange Network ................................. 19
2.4 Partitioning the Multistage Cube Network ....................................... 19
2.5 Summary ................................................................................... 27
3. Simulation Modeling and the Performance of ASEN ............................. 28
  3.1 Introduction ........................................................................... 28
  3.2 Simulating with SLAM II .......................................................... 28
  3.3 Modeling Assumptions .............................................................. 29
  3.4 Modeling the Networks ............................................................. 31
    3.4.1 The Stage Partitioned MSC .................................................... 31
    3.4.2 The MSC with Randomly Grouped PEs .................................... 32
    3.4.3 The Single-Stage Cube with Adaptive Routing ......................... 33
  3.5 Normal Distribution (or Hot-Spots) ............................................. 36
  3.6 Performance Analysis of the ASEN ............................................. 38
    3.6.1 Modeling ASEN with SLAM ................................................... 38
    3.6.2 Throughput Comparison for ASEN and MSC ........................... 39
    3.6.3 Delay Variance ................................................................... 43
    3.6.4 Implementation Cost Analyses .............................................. 43
  3.7 Summary ................................................................................ 44
4. Performance of the Partitioned Multistage Cube Network ................... 47
  4.1 Introduction ............................................................................. 47
  4.2 Comparison of Input and Output Partitioning .................................. 48
  4.3 Comparison of MSCPS and MSCRG ........................................... 57
  4.4 Comparison of Partitioned Packet-Switched and Circuit-Switched MSCs ......................................................... 73
Table of Contents
## List of Figures

<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Flynn's Classification Taxonomy</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Processor-to-Memory System Architecture</td>
<td>15</td>
</tr>
<tr>
<td>2.3 PE-to-PE System Architecture</td>
<td>16</td>
</tr>
<tr>
<td>2.4 A Single-Stage Cube Network with N=8, n=3</td>
<td>18</td>
</tr>
<tr>
<td>2.5 The Four Settings of a 2-by-2 Switch Box</td>
<td>20</td>
</tr>
<tr>
<td>2.6 A Multistage Cube Network with N=16</td>
<td>21</td>
</tr>
<tr>
<td>2.7 An Augmented Shuffle Exchange Network with N=16</td>
<td>22</td>
</tr>
<tr>
<td>2.8 Examples of Partitioned MSC Networks</td>
<td>24</td>
</tr>
<tr>
<td>2.9 An MSC of Size 16 Partitioned into 3 Subsystems</td>
<td>25</td>
</tr>
<tr>
<td>3.1 An Optimal Path from 0001 to 1010</td>
<td>35</td>
</tr>
<tr>
<td>3.2 Average Time in System versus Network Loading Factor (4-by-4 switch size)</td>
<td>41</td>
</tr>
<tr>
<td>3.3 Average Time in System versus Network Loading Factor (16-by-16 switch size)</td>
<td>42</td>
</tr>
<tr>
<td>4.1 Input Partitioned MSC</td>
<td>49</td>
</tr>
<tr>
<td>4.2 Output Partitioned MSC</td>
<td>50</td>
</tr>
<tr>
<td>4.3 Mixed Partitioned MSC</td>
<td>51</td>
</tr>
<tr>
<td>4.4 Performance of MSCPS (64 PEs per Group)</td>
<td>52</td>
</tr>
<tr>
<td>4.5 Performance of MSCPS (32 PEs per Group)</td>
<td>53</td>
</tr>
<tr>
<td>4.6 Performance of MSCPS (16 PEs per Group)</td>
<td>54</td>
</tr>
<tr>
<td>4.7 Performance of MSCPS (8 PEs per Group)</td>
<td>55</td>
</tr>
<tr>
<td>4.8 Performance of MSCRG (32 and 64 PEs per Group)</td>
<td>58</td>
</tr>
<tr>
<td>4.9 Performance of MSCRG (8 and 16 PEs per Group)</td>
<td>59</td>
</tr>
</tbody>
</table>
4.10 An Example of Packet Routing and Switch Contentions in MSCRG.............................. 61
4.11 An Example of Packet Routing in the MSCPS.................................................................. 62
4.12 Performance of MSCPS and MSCRG (64 PEs per Group).................................................. 63
4.13 Performance of MSCPS and MSCRG (32 PEs per Group).................................................. 64
4.14 Performance of MSCPS and MSCRG (16 PEs per Group).................................................. 65
4.15 Performance of MSCPS and MSCRG (8 PEs per Group)................................................... 66
4.16 Performance of MSCPS and MSCRG (64 PEs per Group - Mixed Distribution).................. 69
4.17 Performance of MSCPS and MSCRG (32 PEs per Group - Mixed Distribution).................. 70
4.18 Performance of MSCPS and MSCRG (16 PEs per Group - Mixed Distribution).................. 71
4.19 Performance of MSCPS and MSCRG (8 PEs per Group - Mixed Distribution).................... 72
5.1 Performance of Two Differently Routed Cubes (Uniform Distribution)............................. 82
5.2 Performance of Two Differently Routed Cubes (Normal Distribution)............................... 83
5.3 A Packet Traveling from Node 0001 to Node 1010.............................................................. 85
A.1 Performance of MSCPS and MSCRG (1024 PEs - 128 PEs per Group)................................. 93
A.2 Performance of MSCPS and MSCRG (1024 PEs - 64 PEs per Group)................................. 94
A.3 Performance of MSCPS and MSCRG (1024 PEs - 32 PEs per Group)................................. 95
A.4 Performance of MSCPS and MSCRG (1024 PEs - 16 PEs per Group)................................. 96
A.5 Performance of MSCPS and MSCRG (64 PEs - 16 PEs per Group).................................... 97
A.6 Performance of MSCPS and MSCRG (64 PEs - 8 PEs per Group)..................................... 98
A.7 Performance of MSCPS and MSCRG (64 PEs - 4 PEs per Group)..................................... 99
B.1 Performance of Two Differently Routed Cubes (1024 PEs - Uniform Distribution).............. 100
B.2 Performance of Two Differently Routed Cubes (1024 PEs - Normal Distribution)............... 101
B.3 Performance of Two Differently Routed Cubes (64 PEs - Uniform Distribution).................... 102
B.4 Performance of Two Differently Routed Cubes (64 PEs - Normal Distribution)..................... 103
List of Tables

<table>
<thead>
<tr>
<th>Tables</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Possible Architectures by Skillicorn’s Taxonomy</td>
<td>10</td>
</tr>
<tr>
<td>3.1 Cost of Buffers for the ASEN and the MSC Networks</td>
<td>45</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Background

Over the 50-year era of electronic computers, computation power and speed have increased dramatically with the help of rapid technological advancements. As the technological speedup limit of the von Neumann uniprocessor system was approached [Von46], computer architects implemented pipelining of execution units in their designs and achieved faster uniprocessor systems. Due to this and other design improvements, the capabilities of computer systems changed from calculating simple mathematical functions to performing highly complex numerical analyses. However, the demand for computational power increased at the same time. Society needed computing power to handle large and complex real-time processes, such as numerical weather forecasting, medical research and diagnosis, aerodynamics and structure analysis, image processing, artificial intelligence, defense systems, etc. To accomplish these tasks, a computer system must run at a real-time speed. The classic uniprocessor von Neumann systems, due to their inherent sequential design, and the pipelined systems are limited to only small real-time processes. To solve real-time processes such as the ones listed above, computer architects have to employ parallel processing systems where many processors work together concurrently. With the advent of VLSI technology, it is economically feasible to construct a concurrent processing system by interconnecting hundreds or even thousands of processors and memory modules.

With the introduction of parallel processing systems, a new research area has emerged: developing and analyzing the performance of interconnection networks (INs). An IN has to allow inter-processor and processor-to-memory communications at the speed of the processors. If the IN does not perform at the speed of the processors, it is impossible to achieve real-time processing capability. In designing an interconnection network, the main aspect that the computer architects
consider is the communication speed or the throughput of the IN. They also need to consider other aspects of the IN, such as control strategy, operation mode, network topology, and switching methodology. These are covered in detail in Chapter 2. Various INs have been designed to link multiple processors and memory modules in parallel processing systems [Fen81]. Each IN has its own advantages and disadvantages, and it is up to the architects to choose the right type of IN for a specific parallel system and its intended applications.

To choose a right IN for their system, computer architects need to analyze the performance of the INs. To do this, architects have used mathematical or statistical methods. However, since the size and the complexity of the network increases as more processors are used in the system, mathematical or statistical analysis can become intractable. For this reason, many multiprocessor system architects use computer simulators to model and test their networks. Also, a simulation tool can supply statistical information that can be used to validate existing mathematical or statistical models. With the use of simulation, architects can easily compare performance of different INs and choose the most suitable IN for the system they are designing.

1.2 Research Goals

This thesis investigates two types of INs, the multistage cube network (MSC) and the single-stage cube network, also known as Cube or Hypercube. For a parallel processing machine with many users, a network should handle transmissions among subgroups of processors quickly. The first goal is to analyze the performance of MSC in a multi-user system. Performance of the stage partitioned MSC for a multi-user parallel processing machine is analyzed, and compared to the performance of the same network when its source-destination pairs are randomly grouped to form the same sized subnetwork. Partitionability of the MSC is covered in Chapter 2. The partitioning capability of the multistage cube network is beneficial in large SIMD and MIMD systems. SIMD and MIMD systems are also discussed in Chapter 2. A static routing scheme that
always route messages along a fixed path makes poor use of the network's bandwidth. The second goal is to study the performance advantage of adaptive routing scheme over the static routing scheme. The performance of the Cube under an adaptive (dynamic) routing scheme is analyzed. This result is compared with the performance of the Cube with static routing [Rai87].

A paper by Ramachandran, Raines, Park and Davis investigated the performance of the augmented shuffle exchange network (ASEN) in a packet switched environment. A brief summary of [RaR92] is given Chapter 3. The augmented links in ASEN are used by [KuR87, KuR89] for fault-tolerance, but in [RaR92], the augmented links are used to implement an adaptive routing scheme to reduce congestion problems. Kumar's research in circuit switched ASEN analyzes the delay characteristics of the ASEN network as well as the effects of faults within it [KuR87]. The results of his research show that the ASEN quantitatively outperforms the MSC under a circuit switched environment in terms of message delay. His research analyzed the fault recovery characteristics of the ASEN. [RaR92] complements Kumar's work by providing a comparison of the ASEN and MSC in a packet switched environment and considers the impact of heavy network congestion.

This thesis uses packet switching, and uniform and normal (non-uniform) distributions to generate source and destination addresses. To analyze the performance of the network, the average time delay incurred by a packet as it traverses through the network is used. The performances of the networks are modeled using the SLAM simulation language. An introduction to SLAM and modeling assumptions are covered in Chapter 3. This thesis uses Raines' previous work [Rai87] for verification and comparison of the simulation results. In his research, Raines uses SLAM to analyze the performance of three different types of INs, the multistage cube network, the single-stage cube network, and the mesh network used in Illiac IV machine. Raines uses a uniform distribution to generate source-destination pairs and packet switching methodology. His analyses
concentrate on non-partitioned multistage cube network and the Cube under a static routing scheme.

1.3 Summary

This chapter introduced the need for multiprocessor systems and their interconnection networks. The limitations of the classic uniprocessor von Neumann systems were presented. To achieve the real-time computation capability, multiprocessor systems were developed. One of the main tasks the multiprocessor system designers face is the design of an interconnection network that will not slow the processing speed of the overall system. This study extends previous work done by Raines and investigates two topologies of interconnection networks: the partitioned MSC and the Cube with adaptive routing.

Chapter 2 presents an overview of parallel processing systems. Classification methodologies of parallel processing systems are presented. Then, a detailed discussion of interconnection networks is presented. Different types of interconnection networks are studied including operation modes, control strategy, switching methodology, and network topology. Finally, the partitioning capability of the MSC and the importance of a partitionable network in SIMD and MIMD machines are presented. Chapter 3 starts with a brief introduction of the SLAM simulation language. A discussion of modeling the networks with SLAM and the modeling assumptions follows. Finally, a performance analysis of ASEN is presented. In Chapter 4, a comparison between partitioning by stage and random grouping of processors is presented. Chapter 5 presents the performance of the Cube under an adaptive routing scheme. Conclusions are presented in Chapter 6.
2. Introduction to Parallel Processing Systems

2.1 Introduction

This chapter presents a brief introduction to parallel processing systems. The basic characteristics that are essential in understanding today's multiprocessor systems and in designing efficient systems are discussed. Section 2.2 presents different methodologies that classify types of parallel processing systems. Section 2.3 presents an overview of interconnection networks. Section 2.4 discusses the importance of a partitionable network in SIMD and MIMD machines.

2.2 Parallel Processing Systems Classification Methodologies

With the emergence of various types of parallel processing systems, a need for an effective classification methodology is evident. A good taxonomy will aid the system architects and the users to differentiate many different multiprocessor systems. Four prominent classification schemes are presented below.

2.2.1 Flynn's Taxonomy

The taxonomy introduced by Flynn in 1966 [Fly66] is the most universally accepted method of classifying computer systems. His taxonomy is a quasi-block diagram approach based on the number of instruction and data streams associated with the system. Stream denotes the sequence of items, instructions or data, that are either executed or operated upon by the processors in the system. Based upon this method, Flynn proposed that a computer system can be placed into one of four broad categories:

- SISD -- Single Instruction stream - Single Data stream
- SIMD -- Single Instruction stream - Multiple Data streams
- MISD -- Multiple Instruction streams - Single Data stream
MIMD – Multiple Instruction streams - Multiple Data streams

Figure 2.1 shows system configurations for the four categories.

The SISD machine is equivalent to the classic uniprocessor von Neumann system. This system has one processor that executes a single stream of instructions with a single stream of data. The MISD machine is characterized by multiple processors which execute multiple instruction streams on a single stream of data. Currently, there is no computer system that is a true MISD machine. Parallel processing systems are categorized as either SIMD or MIMD machines. The SIMD machine has multiple processors, each of which has its own data stream, executing a single instruction stream. The SIMD machine operates in a lock-step mode to synchronize the execution done by each processor. A good example of an SIMD machine application is image smoothing where every processor smoothes its segment of a larger image. The MIMD machine has multiple processors executing their own instruction streams with their own data streams. The MIMD machine is characterized as a true parallel machine. Multiple processes are operated independently in an MIMD machine by executing multiple independent instructions with multiple independent data streams.

Flynn's taxonomy is widely accepted due to its compact notation and its ease of classifying any system. However, a disadvantage of this taxonomy is its inability to directly compare different systems that fall within the same class.

2.2.2 Feng's Taxonomy

The taxonomy proposed by Feng [Fen72] compares systems in terms of their degrees of parallelism. His taxonomy attempts to quantify the processing power of a system. Feng's definition of degree of parallelism is the maximum number of bits that can be altered by the execution of one instruction. To measure this degree of parallelism, a cross product of the system's basic word length, \( n \), and its bit-slice depth, \( m \), is computed. An ordered pair \( (n, m) \) describes a system's parallelism and is used in comparing different multiprocessor systems. Thus, a system
Figure 2.1 Flynn's Classification Taxonomy
(a) SISD; (b) MISD; (c) SIMD; (d) MIMD
may be classified as word serial or parallel (n=1 or n>1) and bit serial or parallel (m=1 or m>1). This taxonomy provides supplemental information to Flynn's taxonomy and enables a more detailed comparison among systems within the same class of Flynn's taxonomy. When not used with Flynn's taxonomy, Feng's classification provides little information on system structure.

2.2.3 Händler's Taxonomy

Another classification taxonomy is proposed by Händler [Hän77]. His taxonomy identifies both the degree of parallelism and the degree of pipelining in a computer system. Pipelining is defined as the system's ability to decompose a process into distinct subprocesses which may be executed in an overlapped manner. Händler's taxonomy describes a system by an ordered triple, \( T(C) \), which contains six independent variables. The definition of \( T(C) \) is given below.

\[
T(C) = < K \times K', D \times D', W \times W' > \tag{2.1}
\]

where

- \( K \): the number of processor control units (PCU)
- \( K' \): the number of PCUs that can be pipelined together
- \( D \): the number of arithmetic logic units (ALU) per PCU
- \( D' \): the number of ALUs that can be pipelined together
- \( W \): the word length of the ALU
- \( W' \): the number of pipeline stages in the ALU

The main significance of Händler's taxonomy is that it includes the concept of pipelining in a classification of computer systems.

2.2.4 Skillcorn's Taxonomy

The taxonomy by David Skillcorn [Ski88] consists of three levels of increasing detail and discrimination. The first level describes an architecture by specifying:

- the number of instruction processors (IPs),
• the number of instruction memories (IMs),
• the type of switch connecting IPs to IMs,
• the number of data processors (DPs),
• the number of data memories (DMs),
• the type of switch connecting DPs and DMs,
• the type of switch connecting IPs and DPs, and
• the type of switch connecting DPs to DPs.

The first level of Skillicorn's classification refines Flynn's taxonomy by expanding the SIMD and MIMD classes into a set of subclasses that capture important existing architectures. The second level of Skillicorn's taxonomy allows further extension by describing whether or not the processors can be pipelined and to what degree. The second level also gives the state diagram behavior of the processors. A third level that describes specific implementation details is also possible.

Table 2.1 shows a set of simple architectures possible under the assumptions that the number of memories are equal to the number of processors for each subsystem. Four different forms of abstract switches that connect functional units together are defined as follows:

• 1-to-1 (1 - 1): A single functional unit of one type connects to a single functional unit of another.
• n-to-n (n - n): The \( i^{th} \) unit of one set of functional units connects to the \( i^{th} \) unit of another.
• 1-to-n (1 - n): One functional unit connects to all \( n \) devices of another set of functional units.
• n-by-n (n x n): Each device of one set of functional units can communicate with any device of a second set and vice versa.

Referring to Table 2.1, Classes 1 through 5 are the dataflow/reduction machines that do not have instructions in the usual sense. Class 6 is the classic von Neumann uniprocessor system. Classes
Table 2.1 Possible Architectures by Skillicorn's Taxonomy [Ski88]

<table>
<thead>
<tr>
<th>Class</th>
<th>IPs</th>
<th>DPs</th>
<th>IP-DP</th>
<th>IP-IM</th>
<th>DP-DM</th>
<th>DP-DP</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>none</td>
<td>none</td>
<td>1-1</td>
<td>none</td>
<td>reduct/dataflow uniprocessor</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>n</td>
<td>none</td>
<td>none</td>
<td>n-n</td>
<td>none</td>
<td>separate machines</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>n</td>
<td>none</td>
<td>none</td>
<td>n-n</td>
<td>n x n</td>
<td>loosely coupled reduct/dataflow</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>n</td>
<td>none</td>
<td>none</td>
<td>n x n</td>
<td>none</td>
<td>tightly coupled reduct/dataflow</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>n</td>
<td>none</td>
<td>none</td>
<td>n x n</td>
<td>n x n</td>
<td>von Neumann uniprocessor</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1-1</td>
<td>1-1</td>
<td>1-1</td>
<td>none</td>
<td>von Neumann uniprocessor</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>n</td>
<td>1-n</td>
<td>1-1</td>
<td>n-n</td>
<td>none</td>
<td>von Neumann uniprocessor</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>n</td>
<td>1-n</td>
<td>1-1</td>
<td>n-n</td>
<td>n x n</td>
<td>Type 1 array processor</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>n</td>
<td>1-n</td>
<td>1-1</td>
<td>n x n</td>
<td>none</td>
<td>Type 2 array processor</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>n</td>
<td>1-n</td>
<td>1-1</td>
<td>n x n</td>
<td>n x n</td>
<td>Type 2 array processor</td>
</tr>
<tr>
<td>11</td>
<td>n</td>
<td>1</td>
<td>1-n</td>
<td>n-n</td>
<td>1-1</td>
<td>none</td>
<td>separate von Neumann unprocessors</td>
</tr>
<tr>
<td>12</td>
<td>n</td>
<td>1</td>
<td>1-n</td>
<td>n x n</td>
<td>1-1</td>
<td>none</td>
<td>separate von Neumann unprocessors</td>
</tr>
<tr>
<td>13</td>
<td>n</td>
<td>n</td>
<td>n-n</td>
<td>n-n</td>
<td>n-n</td>
<td>n x n</td>
<td>separate von Neumann unprocessors</td>
</tr>
<tr>
<td>14</td>
<td>n</td>
<td>n</td>
<td>n-n</td>
<td>n-n</td>
<td>n-n</td>
<td>n x n</td>
<td>loosely coupled von Neumann</td>
</tr>
<tr>
<td>15</td>
<td>n</td>
<td>n</td>
<td>n-n</td>
<td>n-n</td>
<td>n x n</td>
<td>none</td>
<td>tightly coupled von Neumann</td>
</tr>
<tr>
<td>16</td>
<td>n</td>
<td>n</td>
<td>n-n</td>
<td>n-n</td>
<td>n x n</td>
<td>n x n</td>
<td>tightly coupled von Neumann</td>
</tr>
<tr>
<td>17</td>
<td>n</td>
<td>n</td>
<td>n-n</td>
<td>n x n</td>
<td>n-n</td>
<td>none</td>
<td>loosely coupled von Neumann</td>
</tr>
<tr>
<td>18</td>
<td>n</td>
<td>n</td>
<td>n-n</td>
<td>n x n</td>
<td>n-n</td>
<td>n x n</td>
<td>loosely coupled von Neumann</td>
</tr>
<tr>
<td>19</td>
<td>n</td>
<td>n</td>
<td>n-n</td>
<td>n x n</td>
<td>n x n</td>
<td>n x n</td>
<td>Denelcor Heterogeneous Element Processor</td>
</tr>
<tr>
<td>20</td>
<td>n</td>
<td>n</td>
<td>n-n</td>
<td>n x n</td>
<td>n x n</td>
<td>n x n</td>
<td>Denelcor Heterogeneous Element Processor</td>
</tr>
<tr>
<td>21</td>
<td>n</td>
<td>n</td>
<td>n x n</td>
<td>n-n</td>
<td>n-n</td>
<td>none</td>
<td>tightly coupled von Neumann</td>
</tr>
<tr>
<td>22</td>
<td>n</td>
<td>n</td>
<td>n x n</td>
<td>n-n</td>
<td>n-n</td>
<td>n x n</td>
<td>tightly coupled von Neumann</td>
</tr>
<tr>
<td>23</td>
<td>n</td>
<td>n</td>
<td>n x n</td>
<td>n-n</td>
<td>n x n</td>
<td>none</td>
<td>tightly coupled von Neumann</td>
</tr>
<tr>
<td>24</td>
<td>n</td>
<td>n</td>
<td>n x n</td>
<td>n-n</td>
<td>n x n</td>
<td>n x n</td>
<td>tightly coupled von Neumann</td>
</tr>
<tr>
<td>25</td>
<td>n</td>
<td>n</td>
<td>n x n</td>
<td>n-n</td>
<td>n x n</td>
<td>none</td>
<td>tightly coupled von Neumann</td>
</tr>
<tr>
<td>26</td>
<td>n</td>
<td>n</td>
<td>n x n</td>
<td>n-n</td>
<td>n x n</td>
<td>none</td>
<td>tightly coupled von Neumann</td>
</tr>
<tr>
<td>27</td>
<td>n</td>
<td>n</td>
<td>n x n</td>
<td>n x n</td>
<td>n x n</td>
<td>none</td>
<td>tightly coupled von Neumann</td>
</tr>
<tr>
<td>28</td>
<td>n</td>
<td>n</td>
<td>n x n</td>
<td>n x n</td>
<td>n x n</td>
<td>n x n</td>
<td>tightly coupled von Neumann</td>
</tr>
</tbody>
</table>
7 through 10 represent the array processors. Classes 11 and 12 are the MISP systems. Finally, Classes 13 through 28 represent various types of multiprocessor systems.

Skillicorn's classification, by extending Flynn's popular classification by increasing the discrimination between different kinds of parallel architectures, captures the distinctions between architectures where they really significantly differ, and reveals underlying relationships as well. His taxonomy also suggests a number of unexplored possible architectures that might be fruitful places to look for new and innovative machine designs. It is more complex than Flynn's work, but is still simple and straightforward enough to be used as a conceptual tool for understanding and as an engineering tool for design.

2.3 Overview of Interconnection Networks

To exploit the real-time computation capability made possible by technological advancements, a parallel processing system must be designed to significantly reduce the overhead for communication between the processors. The communication architecture of the system might support one application well but might prove inefficient for others. Thus, it is up to the computer architect to choose an IN, among many different topologies of INs, that is efficient, reliable, and cost effective for the system being designed.

2.3.1 Traditional Network Designs

The simplest interconnection network is the bus or ring structure. In the bus or ring structure, a common communication path, either unidirectional or bi-directional, is shared among the processors of the system. The complexity of the bus or ring structure is $O(N)$, where $N$ is the number of processors connected to the network. The communication time is also a function of the number of processors in the network. Therefore, the structure can saturate quickly with a relatively low number of processors. This phenomenon limits the bus or ring structure to a system with only a small number of processors.
A network at the other end of the complexity scale is the crossbar switch. The crossbar switch has $N$ inputs and $N$ outputs and provides a direct connection between every input and output of the network. The network is non-blocking except for two inputs requesting the same output concurrently. The crossbar switch is extremely fast -- 1 stage or level of delay -- but it is also complex. The complexity of the crossbar switch is $O(N^2)$. This complexity limits the crossbar switch to a system with a small number of processors.

The limitation of bus or ring structure and crossbar switch to a small size system led architects to find a reasonable compromise in terms of cost and speed between the crossbar and the bus or ring. To reach this compromise, researchers developed two classes of INs that are suitable for systems with a large number of processors. The two classes of INs are the single-stage (direct) networks and the multistage (indirect) networks. A single-stage network is composed of a single stage of switching elements cascaded to a link connection pattern. The single-stage network is also called a recirculating network because data items may have to recirculate through the single stage several times before reaching their final destinations. In a single-stage network, point-to-point links are used to connect the processors. A multistage network consists of more than one stage of switching elements and are capable of connecting an arbitrary input terminal to an arbitrary output terminal. An input terminal and an output terminal can both be connected to processors or connected to a processor and a memory module.

2.3.2 Network Design Decisions

In selecting an architecture for an interconnection network, computer architects must make four design decisions:

- Operation mode
- Control strategy
- Network topology
- Switching methodology
Two types of operation modes can be identified: *synchronous* and *asynchronous*. Synchronous control techniques are well understood and widely used in computer system designs. They are characterized by the existence of a central, global clock that broadcasts clock signals to all devices in a system so that the entire system operates in a lock-step fashion. Asynchronous techniques, on the other hand, operate without a global clock. The communications among operational units in the system are performed by means of interlock hand shaking. As a result, they have good expandability and modularity, but are difficult to design.

A typical interconnection network consists of a number of switching elements and interconnecting links. Interconnection functions are realized by properly setting control of the switching elements. There are two strategies to perform this function: *centralized control* and *distributed control* schemes. In a centralized control scheme, all the control signals come from a single, central controller. In contrast, individual elements within the network control their own functioning in a distributed control scheme.

Two routing schemes for the interconnection networks are *static* routing and *dynamic* routing. In static routing, links between two processors are passive, dedicated buses that cannot be reconfigured for direct connections to other processors. Links in dynamic, or adaptive, routing can be reconfigured by setting the network's active switching elements.

The four major switching methodologies are *circuit switching*, *packet switching*, *virtual cut through*, and *wormhole routing*. In circuit switching, a physical path is established between a source and a destination prior to the transmission of data. This dedicated path is held until data transmission is completed and can cause other paths to be blocked, leading to increased network congestion. In packet switching, data is put in a packet, along with routing information, and routed through the interconnection network without establishing a physical path. In virtual cut through, the packet headers are examined to determine the next appropriate channel for the packet to be transmitted on. Virtual cut through uses a store and forward method of transmitting the packets.
once the packet header has been examined. If blocked, the packet is buffered until the blockage has been resolved [KeK79]. Wormhole routing uses the same basic approach of examining the packet header as virtual cut through. However, the two methods differ in how the packet is handled when a blockage is encountered. Instead of buffering the packet as done by virtual cut through, wormhole routing keeps the packet moving in the network until the blockage is resolved [Dal86].

2.3.3 Machine Architecture (Network's Viewpoint)

From the network's point of view, parallel processing systems can be organized in two ways: a processor-to-memory (P-to-M) system and a processing element-to-processing element (PE-to-PE) system. A processing element (PE) is a processor-memory pair. Figures 2.2 and 2.3 show the P-to-M and PE-to-PE systems. In a P-to-M system, processors are connected to the memory modules through the use of a bi-directional interconnection network. The network is used to fetch instructions (in MIMD) and data (in SIMD and MIMD) from the memory modules by the processors. The processors also use the network for inter-processor communication through common memory locations. The network can be heavily utilized. In a PE-to-PE system, each processor and local memory pair (PE) is connected to an input and an output port of a unidirectional network. The network provides inter-PE communication only since each processor fetches its instructions and data from its own local memory. The network loading of a PE-to-PE system tends to be less than the loading of a P-to-M system.

2.3.4 Examples of Interconnection Networks

Various types of interconnection networks have been developed, studied and sometimes implemented in systems by researchers and designers. Examples of single-stage networks are the mesh (Illiac IV) network, the PM21 (Plus-Minus 21), the shuffle-exchange network, and the single-stage cube network. The augmented data manipulator network (ADM), the omega network, the augmented shuffle exchange network (ASEN), and the multistage cube network (MSC) are all
Figure 2.2 Processor-to-Memory System Architecture
Figure 2.3 PE-to-PE System Architecture
multistage networks. All of these networks and others are discussed in detail in tutorial books [Sie90] and [Hwb84]. In the following subsubsections, the single-stage cube network, the multistage cube network, and the augmented shuffle exchange network are introduced.

2.3.4.1 The Single-Stage Cube Network. The single-stage cube network, also known as Cube [Sie77], is a single-stage network whose dimension, $n$, is determined by the number of processing elements (PE) in the cube. For example, with $N$ processing elements, the dimension, $n$, is $\log_2 N$. Its name, cube, is derived from the physical connection pattern of an $N=8$, $n=3$ network. In an $n$-dimensional cube, the processing elements and switching elements are located on the vertices of the Cube. Each processing element is linked to $n$ adjacent processing elements. For an $n$-dimension Cube, the network uses $n+1$-by-$n+1$ switching elements. Figure 2.4 shows a Cube network with $n=3$. A Cube network consists of $n$ interconnection functions, $Cube_i$:

$$Cube_i(P_{n-1}, P_{n-2}, \ldots, P_1, P_0) = (P_{n-1}, P_{n-2}, \ldots, P'_i, \ldots, P_1, P_0); \quad 0 \leq i \leq n$$ (2.2)

where $P_i$ denotes the $i^{th}$ bit of a PE's address and the prime, $'$, denotes the bit complement operation. The $Cube_i$ function connects a particular PE whose address is given on the left side of the equation to a PE with the address represented on the right side of the equation. The two PEs' addresses differ only in the $i^{th}$ bit position.

2.3.4.2 The Multistage Cube Network. The multistage cube network (MSC) [SiM81, Sie90] is based on the $Cube_i$ function presented above. The MSC consists of $n=\log_p N$ stages, where $N$ is the number of processing elements in the system and $p$ is the size of the crossbar switching element being used to construct the network. Each stage in the network contains $N/p$ switching elements. The switching elements of the $i^{th}$ stage implement the $Cube_i$ function. At stage $i$, the address lines that differ in the $i^{th}$ bit position are paired together in a switching element. The output stage is numbered as 0 and the input stage is numbered as $(\log_p N) - 1$. The switching
Figure 2.4 A Single-Stage Cube Network with $N=8$, $n=3$
element is a device (or a crossbar) whose function is to interchange its \( p \) inputs and \( p \) outputs. For example, a 2-by-2 switch box has four allowable settings: straight, exchange, upper broadcast, and lower broadcast, as shown in Figure 2.5. Figure 2.6 shows an MSC for \( N=16 \) implemented with 2-by-2 crossbar switching elements.

2.3.4.3 The Augmented Shuffle Exchange Network. The augmented shuffle exchange network (ASEN), first proposed by [KuR87], is a variation of the MSC. Both networks require the same number of stages of switching elements, \( n=\log_p N \), and \( N/p \) switching elements per stage where \( N \) is the number of the processing elements in the network and \( p \) is the basic switch size. The principal differences in the two networks are in the switch sizes used, the fault-tolerant capabilities, and the routing of messages. The switch sizes used in implementations of the ASEN and the MSC for a given network size differ by one input and one output, except in the output stage where they use the same switch size. For a multistage cube network design with \( N \) processing elements and switches of size \( p \)-by-\( p \), a corresponding ASEN of size \( N \) would be implemented by using switches of size \( p+1 \)-by-\( p+1 \). The ASEN uses the extra switch inputs and outputs as redundant links within a stage of switching elements to reroute messages when a faulty switch or link is encountered. Messages are redirected in a cyclic manner within the stage preceding the stage where the fault is detected to provide alternate paths that avoid congested or failed components. Figure 2.7 shows a 16-by-16 ASEN.

2.4 Partitioning the Multistage Cube Network

An MSC and the Cube can be partitioned into independent subnetworks of varying sizes [SiM81]. The advantages of partitioning a system are:

- Increased fault tolerance
- Support of multiple users
- Easier program development
Figure 2.5  The Four Settings of a 2-by-2 Switch Box
Figure 2.6 A Multistage Cube Network with $N=16$
(2-by-2 switching elements)
Figure 2.7 An Augmented Shuffle Exchange Network with $N=16$
Machine size matched to task requirements
Independent subtask execution in parallel

A network of size \( N = 2^N \) is partitionable if the network can be divided into independent subnetworks of size \( M = 2^M \), where \( M < N \). Each subnetwork of size \( M < N \) must have all of the connection properties of a MSC built to be a size \( M \). Also, each subnetwork cannot communicate with other subnetworks so that the presence or absence of congestion, queuing problems, etc. that exist in one subnetwork do not affect other subnetworks. Consider grouping PEs into even and odd numbered subgroups for a MSC with 8 PEs. Stage 0 is the only place where even and odd PEs can exchange data. By forcing Stage 0 to the straight connection, partitioning of even numbered PEs and odd numbered PEs occurs. This is shown in Figure 2.8(a). Other examples of partitioning are shown in Figure 2.8(b), (c), and (d). A general rule for partitioning is as follows:

- To create a subnetwork of size \( M \), addresses of the PEs in the partition must have the same value in \( n - m \) address bit positions.
- Partitioning can be implemented by forcing the stages that correspond to the \( n - m \) bit positions to the straight setting.

The partitioning properties of the MSC could be used in a test-bed system in several ways [SiM81]. These include

1. separating the data input source simulator from the data processing system simulator,
2. allowing multiple SIMD and/or MIMD systems of varying sizes to be simulated, and
3. reducing the chance of conflicts occurring in the network.

As an example of 1, one partition could simulate the data input device and another could emulate the data processor. These two partitions would communicate through the MSC network under program control to simulate the interaction that takes place in the real system. As an example of 2, consider Figure 2.9. Here an MSC of size 16 is divided into three subsystems, one MIMD subsystem with 2 PEs, one SIMD subsystem with 4 PEs, and one SIMD subsystem with 8
Figure 2.8 Examples of Partitioned MSC Networks
(a) partition based on the low-ordered bit position
(b) partition based on the middle-order bit position
(c) partition based on the high-order bit position
(d) partitioned into three subnetworks
Figure 2.9 An MSC of Size 16 Partitioned into 3 Subsystems
PEs. The remaining two PEs can be used as the control units for the two SIMD subsystems. Dividing a multiprocessor system into a few smaller subsystems simply requires partitioning of the network. No special control unit has to be assigned. The operating system can be used to force subnetwork independence. This allows a system to be a multi-user system, in which each user has an independent subsystem.

Partitioning can reduce the chance of conflicts in the network. A conflict occurs when both inputs of a switching element want to connect to the same output. To see an example of 3, assume that a system being emulated includes two pairs of PEs, where the processors in each pair communicate frequently. Assume that N=8 and that the PEs paired were 1 with 6 and 5 with 7. PE 1 cannot communicate with 6 while 5 communicates with 7. PE 1 needs the "1-5" switching element set to exchange, while PE 5 needs it set to straight. If the PEs selected were, for example, 0 with 1 and 2 with 3 by following the partitioning rules, then a conflict will not occur.

Another way of dividing an MSC, rather than partitioning, is randomly grouping the PEs to form subgroups. This is done by combining idle PEs into a subgroup of the required size as the user requests. For example, when a user requests to use only four out of 16 PEs and another user requests to use eight PEs, the system controller will randomly pick any four and any eight idling PEs and create two subgroups. Two subgroups may consist of (1, 15, 8, 12) and (0, 3, 4, 7, 9, 10, 13, 14) PE addresses. Randomly grouping the PEs is simpler than partitioning since it does not follow any rule. However, when the PEs are divided by random grouping, the network is not divided into several independent subnetworks. A fault in the network or a conflict affects not only one subgroup of PEs but may affect all the subgroups which should result in extra delays for packets to reach their destinations. This thesis uses simulation models of both the partitioned and randomly grouped MSC networks and analyzes the performance of both networks.
2.5 Summary

An introduction to parallel processing systems was presented in this chapter. Four taxonomies that are used to classify parallel processing systems were presented. Flynn's taxonomy is very widely accepted because of its compact notations and ease of use, but is unable to provide structural details of a system. Taxonomies by both Feng and Händler provide architectural detail of a system, but fail to provide enough information to accurately describe a system. Skillicorn's taxonomy is the most detailed one of the four. His classification shows the distinctions between architectures where they differ significantly and reveals underlying relationships.

The traditional networks, bus or ring and crossbar, were discussed along with their limitations which led to the design of interconnection networks that compromise between the bus or ring and the crossbar. Network design decisions and machine architectures from the network's viewpoint were discussed. Three examples of INs, the Cube, the MSC, and the ASEN, were presented and benefits of partitionable networks were discussed.
3. Simulation Modeling and the Performance of ASEN

3.1 Introduction

This chapter presents the methodology used in the modeling and the simulation of the multistage cube network (MSC) and the single-stage cube network (Cube). It also briefly outlines a performance study of ASEN [RaR92]. Section 3.2 introduces the SLAM II simulation language. Section 3.3 presents modeling assumptions used to simulate the networks. Section 3.4 discusses how the networks are modeled using SLAM. Section 3.5 discusses the need of normal distribution for address generation in an interconnection network simulation and outlines the results of the pilot studies of normal distribution which is used in [RaR92] and this thesis. A brief outline of the performance study done by [RaR92] is given in Section 3.6. Section 3.7 gives a summary of the chapter.

3.2 Simulating with SLAM II

The simulation language used in modeling the MSC and the Cube is the Simulation Language for Alternative Modeling (SLAM), developed by Pritsker and Associates, Inc. [Pri86]. Computer architects have simulated interconnection network models by using high-order languages such as FORTRAN or C. Unlike these simulations, which consists of thousands of lines of codes, a simulation tool like SLAM reduces complexity of the model and the time required in simulating the networks. Simulation codes written in SLAM are compact but complete, making SLAM a good choice for network simulation. SLAM also supports subroutines that are written in FORTRAN. This usefulness of SLAM has already been proven by previous studies, [Rai87, McH90, Ram92, RaR92]. Since the underlying simulation engine is known to be correct, a user need only validate the network model itself and not the simulator. SLAM automatically keeps
track of packets as they traverse through the simulated network. This gives SLAM the capability of providing snap shots of data and/or statistics at user-defined intervals.

SLAM is an event driven simulator and can be used to simulate both discrete and continuous event models. A user can study the simulation results through three reports that are generated by SLAM. The three are summary, echo, and trace reports. The summary report is the primary output from SLAM. The echo and trace reports are additional outputs that can be obtained by the user. The summary report provides statistics on the files, the activities, and/or the variables of the model. The echo report shows the input data and the initial values set before the execution. The trace report provides a snap-shot view of the network at each instance of time when an event occurs. The trace report is valuable in debugging and validating a model.

SLAM provides an entity which is used to model a message or a packet of information as it traverses the network. A set of attributes associated with each entity are used to distinguish one entity from the other. These attributes can be used to represent source addresses, destination addresses, packet lengths, output channels, queue numbers, or other user-defined values. SLAM uses files to keep track of the entity and its attributes as they flow through the network. A file can represent a queue, which stores groups of entities, or a channel. The basic concept of the simulation using SLAM is to have the entities generated at a user-defined rate and then flow through the network, following the paths determined by the user. Each entity that reaches the final destination is terminated. Statistics associated with the entity may be collected when specified by the user.

3.3 Modeling Assumptions

Before the actual simulation models of the MSC and the Cube are created, several operating conditions of the networks and simulation assumptions must be determined. Based on
the operating assumptions used in previous studies [DiJ81, Rai87] and the discussions made in Chapter 2, the network operating conditions and the simulation assumptions are listed below.

1. Both networks are assumed to be operating in a MIMD environment.

2. A PE-to-PE architecture is assumed.

3. Packet switching is used as the method for inter-PE communications. Message buffers are employed at the switches for the storing and forwarding of packets to and from the switches.

4. Messages are assumed to be a single packet in length.

5. Message buffers are assumed to be infinite in length. Packets entering these buffers are transmitted on a first-in-first-out (FIFO) basis.

6. A distributed control scheme is used. Each switching element within the network decides which output link a packet has to take.

7. A Poisson process controls the message interarrival times.

8. The source addresses are generated according to a uniform distribution while the destination addresses are generated according to a uniform or normal distribution over the range of values specified by \( N \), the number of PEs in the network.

9. The normal distribution is defined to have a standard deviation of 0.25\( N \) about a chosen mean. The standard deviation is chosen in this way to ensure hot-spots are generated while maintaining the full range of destination addresses.

10. The unit of measure to determine the average message delay and throughput of the network is the packet cycle time. This is the time that it takes a packet to move from the front of a buffer at a switch and arrive at the buffer of the next switch in its routing scheme. In this thesis, the packet cycle time is normalized to 1 unit.

11. The time delay a packet encounters in a switching element is assumed to be negligible.
12. Network outputs can process messages faster than the messages can be generated. This insures that the output device will not be a bottleneck.

13. Network loading factors range from 3 to 100 percent. A 100 percent loading factor is equivalent to a mean Poisson packet generation rate of one packet every $1/N$ time units.

3.4 Modeling the Networks

The simulation models of the MSC and the Cube used in this thesis are based on the earlier models by Raines [Rai87]. The MSC and the Cube models have been validated against previously published studies [KrS83, DJJ81, AbP86]. The models have been also shown to be highly accurate in terms of their descriptive power [ShD94]. Therefore, these models are good bases on which to develop new models for the partitioned MSC and the Cube with adaptive routing. For the new models, only the models for partitioning PEs and the new routing algorithm need to be developed and validated. The basic underlying models of the networks do not change from [Rai87]. The validation of the new models is accomplished by the use of simulation traces inherent to SLAM. These traces provides statistical data on specified packets and queues as well as overall network performance.

3.4.1 The Stage Partitioned MSC

The address generation routine for the partitioned MSC is different from the routine for the non-partitioned MSC of [Rai87]. For a non-partitioned MSC of size eight (8), the built-in uniform and normal distribution functions generate addresses between, and including, 0 and 7. If the network is partitioned, for example, in stage 0, the built-in distribution functions need to generate destination addresses among (0, 2, 4, 6) or (1, 3, 5, 7). Both of the built-in distribution functions of SLAM are not capable of doing this. To overcome this problem, the address generation routine allows the distribution function to generate a pseudo-destination address between 0 and 3, and then
maps the generated address to (0, 2, 4, 6) or (1, 3, 5, 7) depending on the source address. If the source address is one of (1, 3, 5, 7), the pseudo-destination address 0 is translated to 1, 1 is translated to 3, 2 is translated to 5, and 3 is translated to 7. The pseudo-destination addresses are translated in the same manner when the source address is one of (0, 2, 4, 6). 0 to 0, 1 to 2, 2 to 4, and 3 to 6.

The translation of the pseudo-destination address to the actual destination address is done by checking the bit(s) that corresponds to the stages that are set to partition the network. This is done by a user-written FORTRAN subroutine that is linked to the simulator. The $i^{th}$ bit of the address corresponds to the $i^{th}$ stage of the network. Assume a MSC of size 16 is partitioned in stages 0 and 2, where stage 0 is the output stage. This creates four subnetworks: (0, 2, 4, 6), (1, 3, 5, 7), (8, 10, 12, 14), and (9, 11, 13, 15). The built-in distribution function generates a pseudo-destination address that ranges from 0 to 3. Assume the function generates a source address of 9 (or 1001 in binary) and a pseudo-destination address 3 (or 11 in binary). Since stages 0 and 2 are used to partition the network, the address generation routine uses the bits of the pseudo-address 3 for bits 1 and 3 of the actual destination address (in this case 1 and 1, so the actual address is now 1112). Now looking at bit 0 and bit 2 of the source address 9, it is clear that bit 0 and bit 2 of the actual destination address are 1 and 0. By combining these two results, the actual destination address is 11 (or 1011 in binary). If the source address is 13 (or 1101 in binary), the actual destination address is 15 (or 1111 in binary). This destination address is returned to SLAM and transmitted with the packet.

3.4.2 The MSC with Randomly Grouped PEs

The address generation routine of the MSC with randomly grouped PEs also uses the pseudo-destination address scheme. The reason for this is the same as the reason given in the earlier section. For example, if the MSC of size 16 is divided into 4 groups (4 PEs per group) by random grouping of PEs, the generated pseudo-destination addresses range from 0 to 3. Before
any packet enters the network, the simulation program randomly groups PE addresses into the number of groups selected by the user. A user-written FORTRAN subroutine is called to do this task. First, the subroutine creates an array, TMPGRP, of size \( N \), where \( N \) is the size of the network. The content of each element in this array is same as its index value. Then, the subroutine uses a built-in random number generator to generate index numbers, XNUM. The subroutine moves the element of TMPGRP(XNUM) to PARTGRP(i,j). PARTGRP is a two-dimensional array of size \( I \)-by-\( J \), where \( I \) is the number of the partition group and \( J \) is the number of PEs per group. Elements (i, 1) to (i, j) of PARTGRP contain the destination addresses of PEs in group \( i \). To avoid an event where an index number is generated more than once by the random number generator, '-1' is stored in the element of TMPGRP once the content has been moved to PARTGRP. After an index number, XNUM, is generated, the subroutine checks the content of TMPGRP(XNUM). If the content is '-1', the subroutine discards XNUM and generates another number.

When a source and a pseudo-destination address are generated, another user-written subroutine is called. This subroutine scans PARTGRP to check to which group the source PE belongs, and saves the group number in a variable, \( k \). The pseudo-destination address is copied to another variable, \( k' \). Now, the element \( (k, k) \) contains the actual destination address which is returned to SLAM.

### 3.4.3 The Single-Stage Cube with Adaptive Routing

The adaptive routing algorithm used in this thesis for the Cube is based on one of the four algorithms developed by [ChS90]. [ChS90] developed the algorithm, designated as \( A1 \) by the two authors, and proved that it works, but did not do a simulation study of the algorithm. They only concentrated on the fault-tolerant aspect of the adaptive algorithm. This thesis implements \( A1 \) to create a simulation model of a dynamically routed Cube, and compares the throughput of the dynamically routed Cube to that of the statically routed Cube. The other three algorithms by
[ChS90] are more fault-tolerant than A1, but are more complicated. A1 is chosen over the other three because, for the purpose of this thesis, the performance analysis of the adaptive routing algorithm, the less complicated A1 is sufficient. A1 attempts to route every message to its destination via an optimal path which is defined as a path of length equal to the Hamming distance between the message's source and destination nodes. The Hamming distance between two cube nodes with addresses $u = u_{n-1}u_{n-2}\ldots u_0$ and $w = w_{n-1}w_{n-2}\ldots w_0$ in a Cube with $2^n$ nodes (n-dimensional Cube) is defined as

$$H(u,w) = \sum_{i=0}^{n-1} h(u_i,w_i),$$

where $h(u_i,w_i) = 1$, if $u_i \neq w_i$, or

$$0, \text{ if } u_i = w_i.$$  \hspace{1cm} (3.1)

Due to the special structure of the Cube, once the source node of a path is given, the path can be described by a coordinate sequence that represents the order of the dimensions in which every two consecutive nodes in a path differ [Gil73]. As shown in Figure 3.1, [0001, 0011, 0010, 1010] is an optimal path from the source node 0001 to the destination node 1010, and can also be represented by a coordinate sequence [2,1,4].

In the static routing algorithm used in the Intel iPSC Version 1 system [Int86], each switching element performs an exclusive-or of the source-destination addresses and then scans the resultant bits, or routing tag, from right to left (bit 0 to bit n-1) until it encounters a logic 1. The outgoing channel to be used is the bit position of the first logic 1 encountered in this scanning. Transmitting the packet along the chosen channel places the packet at a node whose address becomes a new source address, where the above process is repeated until the new source is equal to the final destination address.

The adaptive routing algorithm, A1, modeled in this thesis takes advantage of the fact that there are exactly $n$ disjoint paths of length less than or equal $k + 2$ from $u$ to $w$, where $u$ and $w$ are two arbitrary nodes in a cube such that $H(u,w) = k$. These paths are composed of $k$ disjoint
Figure 3.1 An Optimal Path from 0001 to 1010
optimal paths of length \( k \), and \((n - k)\) disjoint alternate paths of length \( k + 2 \) [SaS88]. A channel, referred as primary channel, that corresponds to a logic 1 in the routing tag will lead to a path with length \( k \), and a channel, referred as spare channel, that corresponds to a logic 0 in the routing tag will lead to a path with length of \( k + 2 \). When the switching element receives a packet, it first checks if the packet has reached its destination. If not, the switching element picks one of the \( k \) primary channels that is not busy. If all the primary channels are busy, then the switching element checks for an idle spare channel. If all the spare channels are also busy, the packet is then queued. The switching element repeats the same sequence of operations when it tries to transmit a packet that has been waiting for its turn in the queue.

When a packet is transmitted via a spare channel, it may be routed around the network in cycles and never arrive at its destination. To prevent this, each packet is sent with an \( n \)-bit vector, called the spare tag. A channel that corresponds to the \( i^{th} \) bit of the routing tag is referred as the \( i^{th} \) channel. When a packet is transmitted through a spare \( i^{th} \) channel, the \( i^{th} \) bit of the spare tag is set to logic 1. Any node that later receives this packet checks the spare tag when the \( i^{th} \) channel is one of the available spare channels. The node does not send the packet through the spare \( i^{th} \) channel if the \( i^{th} \) bit is set to a logic 1 in the spare tag even if this channel is not busy. However, if this \( i^{th} \) channel is one of the primary channels and is not busy, the packet may be transmitted through this channel. This routine lets a packet take a particular channel only once when this channel is one of the spare channels in order to prevent a packet from being routed in endless cycles.

3.5. Normal Distribution (or Hot-Spots)

In simulating interconnection network message traffic, the uniform distribution is used to generate destination addresses to estimate the upper limits of network performance. If the distribution of destination addresses is not uniform, some parts of the network will be more
congested than others, contributing to a decrease in throughput. Therefore, using the uniform distribution leads to an analysis of the maximum performance that a network can support [DiJ81]. In addition to the uniform distribution, this thesis uses the normal distribution to generate destination addresses. Even though the uniform destination address distribution gives the maximum throughput of a network, it is not common in real world applications where one or few destination PEs may be more favored by the source PEs. To analyze the network performance in a more practical situation, the normal distribution is also used to generate destination addresses or hot-spots.

Hot-spots, as summarized by [DeC89, PrN85], cause a disproportionate number of packets to be routed to a subset of the destinations. This is done by generating packets over a range of destination addresses with a preferential concentration around a mean destination address. This causes the switch buffers along the tree of paths from the sources to the hot-spot destination to fill quickly. At higher loading factors, the rate of generation is greater than the rate at which packets are removed from the buffers, ultimately causing the network to saturate. All network messages that must cross the tree are affected. By introducing hot-spots in the networks, this thesis analyzes the performance the networks under this real world situation.

For the simulations with hot-spots, packet destinations are chosen using a normal distribution with a specified mean and standard deviation. In all the simulation results in this thesis, the mean is chosen randomly by the user. However, the value chosen for the standard deviation is one-quarter of the network size, based on the results of pilot studies. When the standard deviation is chosen to be less than one-quarter of the network size, some PEs were never chosen to be a destination. When the standard deviation is chosen to be much greater than one-quarter of the network size, the range of the PE addresses favored by the address generator were too wide and did not qualify as a good model of hot-spots. When the standard deviation is about a quarter of the network size, the address generator created a distribution that covered the whole
range of addresses, from \(0\) to \(N-1\), with a concentration of addresses narrow enough to be modeled as a hot-spot.

3.6. **Performance Analysis of the ASEN**

This section presents a performance analysis [RaR92] of the ASEN, which was described in Chapter 2. The author of this thesis contributed in completing [RaR92] by preparing the normal distribution scheme and aiding the analysis of the simulation results. As discussed in Chapter 2, the ASEN was first developed by [KuR87] to be used as a fault-tolerant network. Kumar's research in circuit-switched networks analyzes the delay characteristics of the ASEN as well as the effects of faults [KuR87]. The results of his research show that the ASEN quantitatively outperforms the multistage cube network in a circuit-switched environment in terms of message delay. His research also analyzed the fault recovery characteristics of the ASEN. [RaR92] complements Kumar's work by providing a comparison of the ASEN and the multistage cube networks in a packet-switched environment and considers the impact of heavy network congestion.

Subsequent sections show that the ASEN provides enhanced packet-switched operation at higher network loading factors than the multistage cube network, as well as lower implementation costs. Section 3.6.1 defines the network operating conditions and simulation assumptions. The performance comparison, in terms of average packet delay time, is presented in Section 3.6.2. The delay variance and implementation cost analyses are presented in Section 3.6.3.

3.6.1 **Modeling ASEN with SLAM**

The basic simulation assumptions and operating conditions are the same as the ones described in Section 3.3. Two additional operating conditions are described in below.

1. When a link blockage due to congestion or a fault is encountered, the switching logic determines if the packet should be buffered or routed through an augmented link within the same stage. In the case of a link fault, the switch which is connected to the faulty
link treats the link as if it is always in use and, therefore, never available. Transmission or reception across the faulty link is not attempted. A switch fault is modeled by assuming that all of its input links are faulty. As a result, no switch in the previous stage will attempt to route a packet to the faulty switch.

2. Packets that have been sent over an augmented link are assumed to have a priority over other packets vying for an output port at the next switch.

The ASEN SLAM models are adaptations of Raines' MSC simulation models [Rai87]. The model development for the ASEN network modifies Raines' MSC crossbar switch model by adding the additional augmented links and priority queues at each switch. The underlying simulation algorithm for the switch does not change. Because the switch model for the MSC is known to be valid, the ASEN model requires only validation of the augmented link portions. The verification of the ASEN switch model and packet routing algorithm is accomplished by the use of the simulation traces inherent to SLAM.

3.6.2 Throughput Comparison between ASEN and MSC

Simulations of the MSC and the ASEN are done using network sizes of 64, 256, and 1024. Results are consistent across all sizes. Only the results for the 256 network size is presented here. Analysis of other network sizes, 64 and 1024, can be found in [Ram92]. Both the ASEN and the MSC are implemented with the same number of stages of switching elements. Therefore, the best possible time (minimum packet delay) through either network is $\log N$, the number of stages. Steady-state delays depend on network loading and the number of available channels and queues. The ASEN switch model contains an additional input queue when the switch is in a stage designed to provide source-destination path redundancy [KuR87]. These additional input queues serve as priority queues for packets which have been blocked within a particular stage due to encountering a switch or link fault, or network congestion.
Figures 3.2 and 3.3 show the average packet time in the system for various loading factors, source-destination distributions, and switching element sizes for a representative network size of $N=256$. For light system loading, less than 25 percent of full loading, the ASEN and the MSC have approximately the same performance. However, as the loading increases, the delay curves tend to diverge. For 4-by-4 switches, the ASEN routes packets with an average delay that is 25 percent less than the MSC at a loading factor of 50 percent, using a uniform source-destination distribution (Figure 3.2). The performance improvement is more dramatic for the same distribution at 60 percent loading, where the ASEN average packet delay is approximately 40 percent less than the MSC. Under a normal source-destination distribution, the ASEN packet delay is approximately 40 percent less than the MSC delay at 50 and 60 percent loading. Figure 3.3 for 16-by-16 switches reflects the same trends seen in Figure 3.2. At loading factors higher than 50 percent, the packet delay performance of the two networks differs from 25 to 40 percent, depending on the source-destination distribution. The ability of the ASEN to perform better than the MSC is attributed to its use of augmented links and priority queues.

The effect of hot-spots resulting from the use of a normal address distribution is also seen in Figures 3.2 and 3.3. As discussed in Section 3.5, hot-spots cause the network to saturate quickly. This can be seen clearly in the MSC simulations for uniform and normal message distributions. The ASEN performance, however, was found to be better than that of the MSC under similar operating conditions. Performance improvement is a direct result of the redundant intra-stage links and the alternate routing paths that they create. These links ease the load on those switches whose buffers tend to fill up by routing packets to other switches in the same conjugate subset. Figures 3.2 and 3.3 show that the degradation in the performance of the ASEN when using a normal message distribution, compared to its uniform distribution performance, is much less pronounced than that of the MSC.
Figure 3.2 Average Time in System versus Network Loading Factor
(256-PEs with 4-by-4 switching elements) [RaR92]
Figure 3.3 Average Time in System versus Network Loading Factor (256-PEs with 16-by-16 switching elements) [RaR92]
As the switch sizes are increased in the ASEN, the delay curves for the uniform and normal distributions tend to diverge at much higher loading factors. The divergence stems from the fact that the total number of augmented links available for use decreases as switch size increases, which reduces the number of individual switches in the network. Despite this divergence, Figures 3.2 and 3.3 show that the ASEN, under a normal distribution, outperforms the MSC of the same size operating under a uniform source-destination distribution. For a given network loading value, the ASEN’s average message delay is less than that of the MSC and has a higher saturation point than the MSC. The ASEN performance is better because the augmenting links allow the hot spot’s congestion tree to be spread over more switching elements and thus delaying the onset of saturation.

3.6.3 Delay Variance

The variance of the average message delay is important in the design of networks supporting real-time systems. The analysis of the delay variance follows directly from the packet delay statistics which are available in SLAM’s trace report. Across various switch sizes, the variance of the average time to traverse a switching element, and, by extension, the entire network, is found to be approximately the same for both the MSC and ASEN networks. The use of the augmented links leads to lower average delay without increases in the variance of the delay. The worst case delay variance in the ASEN is for a 2-by-2 switch implementation and is 1.42 time units (the minimum delay encountered by a blocked packet is one time unit). The corresponding variance for the MSC is 1.46 time units.

3.6.4 Implementation Cost Analyses

Just as important as the network delay is the cost of implementing a chosen network. Network costs are a function of the switches, wiring and queues required to construct the network. A cost comparison between the MSC network and the ASEN is considered here for the following
conditions [RaR92]. First, space in an input queue at any switch is defined in terms of unit packet lengths, i.e., a queue capable of storing 5 packets equates to 5 buffers. Second, the network cost is dominated by the total number of buffers used to implement the network queues. Finally, one buffer equates to unit cost. Table 3.1 shows the cost associated with implementing both the ASEN and the MSC for various network and switch sizes. For any network size and basic switching element size combination, the ASEN implementation requires fewer buffers and can, therefore, be expected to cost less than a similar implementation using the MSC. Implicit to Table 3.1 is the length of each switch queue. Analysis of the network packet delays in steady-state show that to achieve a queue overflow rate of less than one percent, the MSC network requires queues with a capacity of six packets while the ASEN requires the queue capacity to be five packets. The $[N/s \times (\log_2 N - 1)]$ augmented queues have a capacity of one packet each. The cost savings of the ASEN can once again be attributed to the augmented links and priority queues used to redistribute packets in a congested environment.

3.7 Summary

This chapter presented techniques used in the simulation models. The SLAM II simulation language was introduced along with its advantages over other simulators written in high-level languages. Modeling and simulation assumptions were outlined and packet address generation using normal distribution was discussed. The modeling algorithms for the MSCs, partitioned by stages and divided by random grouping, and the Cube with an adaptive routing were presented. The need of normal distribution in network modeling and simulation was presented along with some pilot study results.

Performance comparisons of the MSC and the ASEN were presented. The ASEN outperformed the MSC for various network sizes, loading factors, source-destination distributions, and switching element sizes. The ASEN's higher performance came without an increase in the
Table 3.1 Cost of Buffers for the ASEN and the MSC Networks
(Uniform Distribution) [RaR92]

<table>
<thead>
<tr>
<th>Size</th>
<th>Network</th>
<th>64</th>
<th>256</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>2x2</td>
<td>MSC</td>
<td>2304</td>
<td>12288</td>
<td>61440</td>
</tr>
<tr>
<td></td>
<td>ASEN</td>
<td>2080</td>
<td>11136</td>
<td>55808</td>
</tr>
<tr>
<td>4x4</td>
<td>MSC</td>
<td>1152</td>
<td>6144</td>
<td>30720</td>
</tr>
<tr>
<td></td>
<td>ASEN</td>
<td>1024</td>
<td>5504</td>
<td>26624</td>
</tr>
<tr>
<td>8x8</td>
<td>MSC</td>
<td>768</td>
<td>Not</td>
<td>Not</td>
</tr>
<tr>
<td></td>
<td>ASEN</td>
<td>648</td>
<td>Used</td>
<td>Used</td>
</tr>
<tr>
<td>16x16</td>
<td>MSC</td>
<td>Not</td>
<td>3072</td>
<td>Not</td>
</tr>
<tr>
<td></td>
<td>ASEN</td>
<td>Used</td>
<td>2576</td>
<td>Used</td>
</tr>
<tr>
<td>32x32</td>
<td>MSC</td>
<td>Not</td>
<td>Not</td>
<td>12288</td>
</tr>
<tr>
<td></td>
<td>ASEN</td>
<td>Used</td>
<td>Used</td>
<td>10272</td>
</tr>
</tbody>
</table>

1 buffer = unit cost

*MSC: 6 buffers per queue; ASEN: 5 buffers per queue

* The network size in question cannot be constructed using this switch element size.
variance of packet delay and with lower implementation cost, i.e., fewer buffers were required. The performance benefits of the ASEN are attributed to the use of augmented links and priority queues. This shows that the ASEN provides a low cost, high performance, packet-switched network for parallel processing.
4. Performance of the Partitioned Multistage Cube Network

4.1 Introduction

Chapter 4 presents the analysis of the simulation results of two different multistage cube networks. These are the MSC partitioned by stage, referred to as MSCPS, and the MSC divided by random grouping of PEs, referred to as MSCRG. Prior to the comparison of MSCPS and MSCRG, the simulation results of Input and Output partitioned MSCs are analyzed. The input and output partitions are two subtypes of the stage partitioned MSCs. They are defined later in the chapter. In modeling these divided MSC networks, three network sizes are considered: N=64, N=256, and N=1024. These network sizes are representative of multiprocessor systems that can be implemented using current microprocessor technology. Analysis of the simulation results show trends that are consistent across various network sizes. Because of this consistency, only results for a representative network of size N=256 are discussed. Results for network sizes of 64 and 1024 are presented in Appendix A. All of the MSCs are constructed using 2-by-2 crossbar switches. Both MSCPS and MSCRG were evaluated using loading values ranging from 3 percent to 70 percent. Note that 100 percent loading is equivalent to N packets entering the network per unit time, where N is the size of the network.

The analysis in this chapter shows that the network performance is primarily dependent on the size of the network, the number of partitioned groups (or the number of PEs per group), the loading factor, and the type of distribution used to generate the destination addresses of the packets. Because of the above independent parameters, it is necessary to vary these factors jointly when comparing the network performance. Among the numerous performance characteristics that are obtainable from a network modeling study, this chapter mainly focuses on the average time a packet takes to traverse the network. First, the average time a packet takes to traverse Input and
Output partitioned networks is analyzed. This is presented in Section 4.2. Section 4.3 compares the average time for MSCPS and MSCRG. Davis and Siegel have studied the circuit-switched, partitioned MSC in [DaS85]. Section 4.4 compares the performance study of the packet-switched partitioned MSC with the performance of the circuit-switched, partitioned MSC studied by [DaS85]. A summary of the chapter is given in Section 4.5.

4.2 Comparison of Input and Output Partitioning

The MSC can be divided into subnetworks by 1) input, 2) output, or 3) mixed partitioning. The input partition puts all of the partitioned stages at the input side of the network. Figure 4.1 shows a two-stage input partitioned MSC of size $N=16$ ($n=4$ stages). Stages 3 ($n-1$) and 2 ($n-1$) are partitioned. The output partition places all of the partitioned stages at the output side of the network. Figure 4.2 shows a two-stage output partitioned MSC of size $N=16$. Stages 1 and 0 are partitioned. The mixed partition uses a random pattern of partitioned and non-partitioned stages. Figure 4.3 shows an example of the mixed partitioned MSC. In the case of circuit switching, the input partition performed slightly better than the output partition [DaS85].

An important parameter of the performance comparison for the MSC is the average time required for a packet to traverse the network. The best possible time (minimum packet delay) through an MSC network is $\log N$, or the number of stages. The steady-state delays depend on network loading, the number of available channels and queues, and the number of partitioned groups.

The dependency of steady-state delays on the network loading and the number of partitioned groups can be seen from Figures 4.4 to 4.7. For higher network loads, the average delay experienced by the packets is higher due to the dependency of steady-state delays on network loading. More packets are present in the network for higher loading and the competition for a link or a switching element among packets is higher. In Figures 4.4 to 4.7, as the number of partitioned...
Figure 4.1 Input Partitioned MSC
(N=16, n=4)

4. Performance of the Partitioned Multistage Cube Network
Figure 4.2 Output Partitioned MSC
\((N=16, n=4)\)
Figure 4.3 Mixed Partitioned MSC
(N=16, n=4)
Figure 4.4 Performance of MSCPS
(256 PEs - 4 Groups with 64 PEs per Group)
Figure 4.5 Performance of MSCPS
(256 PEs - 8 Groups with 32 PEs per Group)
Figure 4.6 Performance of MSCPS
(256 PEs - 16 Groups with 16 PEs per Group)
Figure 4.7 Performance of MSCPS
(256 PEs - 32 Groups with 8 PEs per Group)
groups (subnetworks) increases, the steady-state delays decrease for both the uniform and the normal address distribution schemes. This is the result of the fact that as a MSCPS (both input and output partitions) is partitioned into more subnetworks, the number of PEs in a subnetwork decreases, which reduces the potential for contention within each subnetwork. Since each subnetwork is composed of independent sets of switching elements and links, packets compete with the packets of the same subnetworks. Also, for smaller group sizes, more stages are partitioned and have no queuing delays associated with contention. Their delay is just 1 unit of propagation delay. Therefore, the packets go through more stages faster.

The dependency of steady-state delays on the size of the network can be observed by comparing the average time in system graphs for all three network sizes (64, 256 and 1024) given in this chapter and in Appendix A. Larger network size means more number of available channels and queues which should minimize the extra delay packets experience. This can be verified by studying the difference between the average delay of the lowest load (3.125 percent) and that of the highest load (70 percent) for each network size. The difference is the smallest for the network size of 1024 and the largest for the network size of 64 which shows the dependency of the average delay on the network size.

It is clear from Figures 4.4 to 4.7 that the output partition performs slightly better than the input partition, but the difference is so small, about or less than one percent, that it is reasonable to say that the performances of the input and the output partitions are virtually the same. The small additional delay experienced by the packets of the input partition is explained by using a six-stage MSC (N=64) with three stages partitioned (8 PEs per group) as an example. The summary reports generated by SLAM for both input and output partitions show that at each partitioned stage, except the input stage (stage n-1 or stage 5 in the example) of the input partition, the queuing delay the packets experience is 0 and the switching delay is 1 (as stated in Section 3.3). The first packet that arrives at any of these partitioned stages is switched straight to the output channel without any
delay in the queue. Other packets that follow are also transmitted to the output channel without any queuing delay. Two packets arriving at two different input channels of a switching element within the stage can be transmitted at the same time since there is no contention. This is true since both channels of the switching element are set to straight.

For the input stage of the input partition, the queuing delay the packets experience is not necessarily zero. The packets can arrive at the input stage at a rate exceeding the rate of transmission by the switching element as generated by the simulator. Thus, there is a likelihood of a packet arriving at network's input before the previous one has been transmitted. Therefore, even though the input stage is partitioned, the packets may not be immediately transmitted upon arrival at the switching element, resulting in a non-zero queuing delay. When $k$ stages are partitioned in MSCPS ($k=3$ in the example), only $k-J$ (2 in the example) stages of the input partition have a zero queuing delay while all $k$ stages of the output partition have a zero queuing delay. This one extra stage with a non-zero queuing delay leads the input partition to have a slightly higher average delay than the output partition. In the next section where the performances of MSCPS and MSCRG are compared, whenever the performance of MSCPS is mentioned, the performances of the input and the output partitions are assumed to be the same.

4.3 Comparison of MSCPS and MSCRG

The performance comparison of the MSC partitioned by stage (MSCPS) and the MSC divided by random grouping of PEs (MSCRG) is also done by analyzing the average time required for a packet to traverse the network. The dependency of steady-state delays on the number of partitioned groups for MSCRGs can be seen from Figures 4.8 to 4.9. Unlike MSCPSs, there are virtually no changes in the average packet delay as the number of subnetworks increases. In MSCRGs, dividing up the network into several subgroups of PEs does not reduce contention for the same switching elements by the packets. MSCRG only divides the PEs into subgroups, and
Figure 4.8 Performance of MSCRG
(256 PEs - 8 Groups with 32 PES per Groups
and 4 Groups with 64 PEs per Group)
Figure 4.9 Performance of MSCRG
(256 PEs - 32 Groups with 8 PES per Group and 16 Groups with 16 PEs per Group)
does not divide the network. Each subgroup of MSCRG can be widely spread out through the network and not independent to the operation of other subgroups. One packet from each subgroup may have to use the same switching element to reach the destination which results in switching element or link contention as illustrated in Figure 4.10. In this figure, one subgroup of four PEs is made up of PEs 0, 1, 4, and 6 and the other subgroup is made up of PEs 2, 3, 5, 7. When source-destination pairs for the packets are selected as (0, 6), (1, 4), (2, 7), and (3, 5) at an instant, there is contention at four different switching elements. In MSCPS with two subnetworks of four PEs each, there is no contention as seen in Figure 4.11.

The MSCRG's dependency on network loading is the same as for MSCPS. As the network load increases, the average delay experienced by the packets increases. More packets are present in the network for higher loading and a packet has to compete more with other packets to reach its destination. Figures 4.12 through 4.15 show direct comparisons between the MSCPS and the MSCRG for different numbers of PEs per group. As the loading factor are increases, the difference between the MSCPS and the MSCRG increases drastically for both uniform and normal distributions. It can be observed from these figures that the difference in the average packet delay between uniform and normal distributions is less in MSCRG than in MSCPS. Also, the difference is fairly constant over the different numbers of PEs per group for MSCRG. In MSCPS, the difference in the average packet delay between uniform and normal distributions decreases as the number of PEs per group increases due to smaller independent subnetworks. As mentioned earlier, the subgroups of MSCRG are not independent of each other as in the case of the subnetworks in MSCPS. The hot-spots, introduced in each subgroup of MSCRG, are randomly spread across the network causing the overall source-destination pair distribution to be close to a uniform distribution. Each subnetwork is mixed with the others within the network so the resulting average packet delay is as if the network is not divided at all.
Figure 4.10 An Example of Packet Routing and Switch Contentions in MSCRG
Figure 4.11 An Example of Packet Routing in the MSCPS
Figure 4.12 Performance of MSCPS and MSCRG
(256 PEs Divided into 4 Groups with 64 PEs per Group)
Figure 4.13 Performance of MSCPS and MSCRG
(256 PEs Divided into 8 Groups with 32 PEs per Group)
Figure 4.14 Performance of MSCPS and MSCRG
(256 PEs Divided into 16 Groups with 16 PEs per Group)
Figure 4.15 Performance of MSCPS and MSCRG
(256 PEs Divided into 32 Groups with 8 PEs per Group)
For the uniform distribution, the MSCRG begins to saturate for a loading factor between 40 to 50 percent depending on the number of PEs per group. However, at this loading factor, the MSCPS continues to route packets through the network without saturation. In fact at 50 percent, MSCPS’s packet delay time is about 7 percent (for 64 PEs per group, Figure 4.14) to 33 percent (for 8 PEs per group, Figure 4.11) lower than that of MSCRG. The MSCPS begins to saturate around a 70 percent loading factor. The MSCPS’s saturation point is 20 to 30 percent higher than the MSCRG’s saturation point which enables the MSCPS to handle more network-intensive tasks.

Similar results may be observed for the normal distribution. From Figures 4.13 through 4.15, it is clear that the MSCPS continues to outperform over the MSCRG even with the hot-spots in the network that the normal distribution can cause. As the size of the subnetwork decreases, the delay curves diverge more dramatically. For the network with 32 PEs per subnetwork (Figure 4.13) at 50 percent loading, the average packet delay for MSCRG is higher than that for MSCPS by 6 percent. In the case of 16 PEs per group (Figure 4.14) at 50 percent loading, the average packet delay time for MSCRG is higher than that for MSCPS by about 13 percent. For 8 PEs per subnetwork, the average packet delay for MSCRG is about 23 percent higher than that for MSCPS (Figure 4.15). Finally, as in the case of the uniform distribution, MSCRG begins saturate for loading between 33 percent and 50 percent while MSCPS saturates at 60 percent to 70 percent, depending on the number of PEs per group.

In the case of 64 PEs per group, the average delay for MSCPS is just slightly higher, only about 1 percent, than that for MSCRG (Figure 4.12). For MSCPS, only two stages are partitioned which gives a slight decrease in the overall input stage to output stage transmission delay. When the subgroups are formed by randomly choosing PEs, the subgroups may have been created in a way that they are similar to the subgroups of MSCPS. Even though many different MSCRGs are created and their performance figures are averaged, the resulting figures came close to that for
MSCPS. The different combinations of PEs for each subgroups could be studied, but it is impractical to do so.

The results in Figure 4.12 might lead to a conclusion that the performance of the MSCPS is not necessarily better than that of MSCRG. The normal distribution occurs in real world application, but the probability of having all of the subgroups using the normal destination distribution is low. In fact, all of the subgroups having the uniform distribution is not common in practice either. This is because each subuser's application will have a different source-destination distribution. To test a more realistic destination distribution, several simulations were studied with half of the subgroups with the normal distribution and the other half with the uniform distribution. The results are shown in Figures 4.16 through 4.19. These figures show that for all subgroup sizes, including 64 PEs per group, MSCPS performs better than MSCRG. Among the three differently partitioned MSCPS, the difference in the performance is so small that it is safe to say that all three different partitioning strategies perform equally.

To summarize, in average time in system comparisons, MSCPS performs better than MSCRG in every case for the uniform and mixed distributions, and in all but one case for the normal distribution. As the size of the subgroups decreases, the performance improves for MSCPS while there is no significant change in the performance for MSCRG. Also, since MSCPS has independent subnetworks for each subgroup, any heavy congestion or any switching element or link failure in one subgroup has no effect in other subgroups. This prevents a network intensive task slowing down the rest of the tasks when MSCPS is used for multi-task operation. Also in MSCPS, when there is a fault at a link or a switching element of a subnetwork making the subnetwork unable to transmit packets, other subnetworks are not affected by this fault and can still transmit packets to their destinations. However, this is not true for MSCRG since the network for MSCRG is same as an undivided MSC network with all the subgroups connected. If there is a fault in the MSCRG's network, all of the subgroups may be affected by the fault, resulting in a
Figure 4.16 Performance of MSCPS and MSCRG
(256 PEs Divided into 4 Groups 64 PEs per Group - Mixed Distribution)
Figure 4.17 Performance of MSCPS and MSCRG
(256 PEs Divided into 8 Groups with 32 PEs per Group - Mixed Distribution)
Figure 4.18 Performance of MSCPS and MSCRG
(256 PEs Divided into 16 Groups with 16 PEs per Group - Mixed Distribution)
Figure 4.19 Performance of MSCPS and MSCRG
(256 PEs Divided into 32 Groups with 8 PEs per Group - Mixed Distribution)
total network failure. From the above analysis, it can be concluded that MSCPS outperforms MSCRG.

4.4 Comparison of Partitioned Packet-Switched and Circuit Switched MSCs

In [DaS85], the performance of partitioned circuit switched MSC is studied. Unlike the packet-switched MSC model, where the time a packet takes to be transmitted from a source to a destination is measured, [DaS85] measures the circuit-switched path set-up time. Davis and Siegel analyzed both the hold conflict algorithm and the drop conflict algorithm. In the case of the hold conflict algorithm, for each subnetwork size and message length evaluated, the input partition had lower set-up time than the output partition. The output partition exhibited set-up times ranging from 5 to as much as 26 percent greater than those of the input partition. The analysis results for the drop conflict algorithm are similar to the hold conflict algorithm. The output partitioned network was seen to give set-up times ranging from 11 to 29 percent greater than those of the input partition.

For both the hold and drop algorithms of the partitioned circuit-switched MSCPS, input partition performed better than the output partition. However, for packet-switched MSCPS, both input and output partitions performed similarly. Although the output partition performed slightly better for the partitioned packet-switched MSC, the differences are so small that the performance of the two partitioning methods can be considered equal. These different results can be attributed to the difference between the two message transmission mechanisms. Finally, similar to the case of packet-switched MSCPS, the circuit-switched MSCPS showed better performance as the number of PEs per group decreased.

[DaS85] also compared the performance of the circuit-switched MSCPS with that of the circuit-switched MSCRG. When the simulation results of the circuit-switched MSCRG are compared to the simulation results of the circuit-switched MSCPS, the randomly grouped MSC
gave substantially higher set-up times than that of both input and output partitioned MSCPS. Additionally, the set-up times for the random groupings were observed to be approximately the same as the set-up times for the general five-stage network given in [LeW84]. Little or no improvement in the set up time was obtained by using the random groupings of the system resources. The performance comparison results between the circuit-switched MSCPS and MSCRG are consistent with the comparison results between the packet-switched MSCPS and MSCRG.

4.5 Summary

In this chapter, the performance of the packet switched-partitioned MSC network was analyzed. First, the performance difference between input and output partitions was studied. The output partition performs only slightly better than the input partition such that it is reasonable to conclude that the two partitions perform equally. When these results are compared with the performance of the randomly grouped MSC, MSCPS greatly outperforms MSCRG. The reason is that each subgroup of MSCPS has its own independent subnetwork which is not affected by heavy congestion of other subnetworks. In MSCRG, a heavy congestion due to one subgroup affects the rest of the subgroups causing extra delay since all the subgroups share one network. Besides having lower average time in system than MSCRG, MSCPS has another advantage over MSCRG. When there is a link or switching element fault in one of the subnetworks, the rest of the subnetworks of MSCPS continue to transmit packets without any extra delay since all the subnetworks are independent to each other. In the case of MSCRG, a link or switching element failure may prevent all the subgroups from transmitting packets.

Finally, work done on the circuit-switched partitioned MSC by Davis and Siegel [DaS85] was briefly outlined. [DaS85] showed that the performance of input partition is better than the output partition in the circuit switched MSC. The different results between the circuit-switched
and the packet-switched partitioned MSCs are due to the different mechanisms used to transmit packets. [DaS85] showed similar results when the performance of MSCRG and MSCPS are compared in circuit-switched environment. MSCPS gives lower average packet delay than MSCRG. Concluding from the results of the analysis done in this thesis and [DaS85], MSCPS outperforms MSCRG in every respect.
5. Performance of the Adaptive Routed Single-Stage Cube Network

5.1 Introduction

Network schemes that always route messages along a static, fixed path make poor use of bandwidth since they may block when alternative paths are available. They are also particularly susceptible to component failures. Adaptive routing schemes can use alternative paths between communicating processors, making more efficient use of network bandwidth and providing resilience to failures. The latter property is particularly important for large-scale architectures, since expanding system size can increase the probability of encountering a faulty network component. Chapter 5 presents the analysis of the simulation results of the two differently routed the Cubes. The two networks are the Cube with adaptive routing, referred to as CubeAR, and the Cube with static routing, referred to as CubeSR. Based on the network sizes that are representative of multiprocessor systems that can be manufactured using current microprocessor technology, three network sizes are considered: \( N=64 \), \( N=256 \), and \( N=1024 \). This chapter discusses only the network of size \( N=256 \) because analysis of the simulation results show trends that are consistent among all three network sizes. The results for network sizes of 64 and 1024 are given in Appendix B. Both CubeAR and CubeSR are simulated with loading values ranging from 3 percent to 70 percent.

For the two networks discussed in this chapter, the network size, the loading factor, and the type of source-destination distribution dictate performances. The performance of the two Cubes are analyzed by varying the above parameters. Among the many performance attributes, this chapter mainly focuses on the average time a packet takes to traverse the two Cubes. Brief presentations of other studies on Cubes with different adaptive routing schemes are given in
Section 5.2. Section 5.3 presents the comparison of the average packet time delay of CubeAR and CubeSR. Section 5.4 gives a summary of the chapter.

5.2 Other Studies on the Performance of the Adaptive Routed Cube

This section presents short summaries of three other studies of the Cube with adaptive routing. Brief outlines of the simulation algorithms and assumptions and the simulation results of these studies are presented in the following subsections. Readers are referred to the original papers for more detailed discussions.

5.2.1 Adaptive Routing Protocols for Hypercube Interconnection Networks

The paper by Gaughan and Yalamanchili presented simulation studies to evaluate the performance of representative progressive and backtracking protocols compared to the oblivious protocol e-cube [GaY93]. Gaughan and Yalamanchili chose to examine the performance of DP (Duato's protocol) and TPB-2 (two-phase misrouting backtracking) versus that of e-cube. In addition, they wished to demonstrate the added performance obtained through the use of virtual channels. e-cube routes the packet one dimension at a time in fixed order, from the highest indexed dimension to the lowest. e-cube is identical to the static routing scheme used in Section 5.2, except for the order of the dimensions that are checked by the switching element. In their simulations, the e-cube is implemented with virtual channels.

A virtual channel is a logical entity associated with a physical link used to distinguish multiple data streams traversing the same physical channel. Virtual channels increase network throughput by reducing physical-link idle time. Increased sharing of network resources provides a dramatic increase in network throughput. The DP protocol is a progressive protocol, while the TPB-2 protocol is a backtracking protocol. The progressive protocols will wait, abort, or misroute, preferring to move forward when at an intermediate node when no link through which the message needs to traverse is free. Backtracking protocols back up and start over at a previous
node. Backtracking protocols search the network for a path in a depth-first manner working on the premise that it is better to be searching for other paths than to be waiting for one to become available. More discussions on virtual channel, DP protocol and TPB-2 protocol can be found in [GaY93 and Dua91].

[GaY93] gives simulation results of e-cube DP and TPB-2 protocols. DP and e-cube are implemented with wormhole routing and TPB-2 is implemented with pipelined circuit switching. Two different e-cubes are simulated one with two virtual channels and the other with four virtual channels. Both DP and TPB-2 use four virtual channels per link. The results show that adding an extra pair of virtual channels per physical channel dramatically increases throughput for the e-cube. Its peak throughput nearly doubles. When the message length is $L=16$ flits, the added overhead of pipeline circuit switching does not justify the use of TPB-2 over e-cube using four virtual channels. However, in the case where $L=64$, the added adaptability of TPB-2 gives increased performance for large network loads. DP combines the low overhead of wormhole routing and the flexibility of adaptive routing to get the best performance overall in both cases.

Gaughan and Yalamanchili concluded that in applications with large message sizes (greater than 64 flits), backtracking protocols are better at higher network loads. In addition, because backtracking protocols can regress and try alternate paths from earlier nodes, they are better protocols for fault-tolerant routing than progressive protocols. For overall performance, however, wormhole routing has the edge, and DP gives performance over a wide range of factors that appears difficult to improve on.

5.2.2 Adaptive Routing and Deadlock Recovery: A Simulation Study

Through simulation, Reeves, Gehringer and Chandiramani studied the performance of three adaptive wormhole routing strategies, and compared them with static routing [ReG89]. The static routing selects the links in dimension order, starting with the least significant dimension in

5. Performance of the Adaptive Routed Single-Stage Cube Network
which the source and destination node numbers differ and progressing toward the most significant. The three adaptive strategies investigated were as follows:

1. Route over the first open link (in dimension order) that takes the message toward its destination. (An "open" link is an idle link; an "open neighbor" is a neighbor linked to the current node by an idle link.)

2. Route to the open neighbor that has the greatest number of outgoing free (non-busy) links. Ties are broken by dimension order (the least significant to the most significant).

3. Route to the open neighbor that has the greatest number of outgoing free links in "useful" dimensions (dimension that will take the message further toward its destination).

To evaluate how well these strategies performed, [ReG89] compared the three with an "ideal" routing network, which simulated how fast messages would arrive if they never experienced contention for routers or links.

All of the adaptive strategies outperformed static routing for all combinations of parameters. In tests with all long messages, the adaptive strategies performed in order of their sophistication, with strategy 1 being the worst and strategy 3 being the best. The maximum improvement from substituting adaptive strategy 3 for static routing was about a 30 percent decrease in gendelay, delay since message generation, and a 25 percent increase in throughput. Even though it is a small margin, when all messages are short, adaptive strategy 1 consistently outperformed strategy 2. Based on their simulation results, Reeves, Gehringer, and Chandiramani concluded that adaptive strategy 3 always outperformed the other strategies for all network dimensions, traffic loads, and conditions. Static routing was worst, and strategies 1 and 2 were somewhere in between.
5.2.3 Adaptive Packet Routing in a Hypercube

As a feasibility study of adaptive routing, Kim and Reed selected four representative adaptive routing methods for testing on the Intel iPSC: NRCC routing, shortest queue routing, delta routing, and hybrid weighted routing [KiR88]. They simulated the four methods and compared the performance with that of the static routing method. In NRCC routing, routing decisions are made at a central node, the Network Routing Control Center. Shortest queue routing sends packets on those links with the shortest queue lengths. This method uses only local information.

Delta routing exploits both global information and local information. A designated central node collects global information and selects multiple, acceptable candidate paths based on global information, and delegates the final routing decisions to local nodes. A parameter $\delta$, whence the name delta routing, coordinates the relative routing responsibility of the central node and local nodes. To avoid choosing poor paths at local nodes, as is possible in delta routing, hybrid weighted routing uses both global and local information in the local path selection procedure. Local nodes use both the global path length information and local queue length information to select paths. The parameter $\kappa$ ($0 \leq \kappa \leq 1$) is the weight given to global information.

The results of the simulations show that the best performance of delta routing is achieved at $\delta=9$. It is clear that good performance is obtained by assigning more responsibility to local information. Similarly, the importance of local information can be seen in the results of hybrid weighted routing simulation. Kim and Reed also simulated the four adaptive routing schemes and the static routing scheme to measure the packet delay as a function of a workload. The packet delay increases sharply when greater credit (larger $\kappa$) is given to global information. The performance of static routing and NRCC routing deteriorates most sharply as the workload increases. The network saturates at a workload of 56 packets per second when static or NRCC routing is used. The other three routing algorithms have almost the same packet delays at low
workloads. However, the performance of shortest queue routing starts degrading most sharply from a workload of 50 packets per second. Two hybrid routing algorithms, delta and hybrid weighted, show the best performance. Hybrid weighted routing performs slightly better than delta routing at workload greater than or equal to 56 packets per second.

5.3 Comparison of CubeAR and CubeSR

The performance comparison of CubeAR and CubeSR is done by analyzing the average time required for a packet to traverse the network. The dependency of steady-state delays on the network loading can be seen in Figures 5.1 and 5.2. As the network load increases, the average delay experienced by packets increases. For higher loading, more packets traverse through the network creating higher contention among packets for channels and switching elements. The dependency of steady-state delays on the number of available channels and queues can be observed by comparing the average time in system for all three network sizes (64, 256, and 1024) given in this chapter and in Appendix B. The difference between the average delay of the lowest load (3.125 percent) and that of the highest load (70 percent) is the smallest for the network size of 1024 and the largest for the network size of 64. This is because for larger networks there are more links and queues for the packets as they traverse through the network. The dependency of steady-state delays on network loading can also be seen in Figures 5.1 and 5.2. As the network load increases, the average delays experienced by the packets increases. This is due to more competition for a link or a switching element among packets as more packets are present in the more heavily loaded network.

Direct comparison between CubeAR and CubeSR can be seen in Figures 5.1 and 5.2. The average delay for CubeAR is consistently lower than for CubeSR. In fact, as the loading factor increases, the delay curve of CubeSR diverges away from the curve of CubeAR. For the uniform distribution results shown in Figure 5.1, CubeSR begins to saturate around a 60 percent loading.
Figure 5.1 Performance of Two Differently Routed Cubes
(256 PEs; Uniform Distribution)
Figure 5.2. Performance of Two Differently Routed Cubes
(256 PEs; Normal Distribution)
factor. However, at this loading factor, CubeAR continues to route packets through the network without saturation. At 60 percent, CubeAR's packet delay is about 8 percent lower than that of CubeSR. CubeAR begins to saturate around a 70 percent loading factor. The 10 percent higher saturation point of CubeAR allows this network to handle network-intensive tasks better than CubeSR. Similar results can be observed for the normal distribution. As shown in Figure 5.2, CubeAR outperforms CubeSR even with hot-spots in the network. CubeSR begins to saturate around 50 percent while CubeAR continues to route packets without being saturated. CubeAR starts to saturate at 60 percent. At 50 percent, CubeSR's packet delay time is about 5 percent higher than that of CubeAR.

CubeAR outperforms CubeSR by using one of the idle primary channels or one of the idle spare channels when the primary channel of the first choice is busy. Consider a packet that is trying to move from 0001 to 1010. As illustrated in Figure 5.3, the packet at node 0001 cannot move to node 0000 since the channel linking the two nodes (indicated by an X) is busy. In CubeSR, because of the routing scheme explained in Section 3.4.3, channel 0001-0000 is the only channel that can be used by the packet at 0001. Since the channel is busy, the packet is queued until the channel is idle. However, in CubeAR, the packet at 0001 is transmitted through either channel 0001-0011 or channel 0001-1001. If above channels are all busy, the packet is transmitted through one of the spare channels. In this example, 0001-0101 is the only spare channel. In the example of Figure 5.3, channel 0001-0011 is not busy so the packet is transmitted through this channel without any queuing delay.

Unlike the packets in CubeSR that have only one choice of channel at any switching element, the packets in CubeAR have several channels to choose from. This increases the chance that packets in CubeAR are transmitted without any queuing delay. This adaptive routing scheme of CubeAR reduces contention for a link or a switching element, which reduces the average delay. Less contention among packets also makes the saturation point of the network occur at higher
Figure 5.3 A Packet Traveling from Node 0001 to Node 1010
loading factor. The example of Figure 5.3 can be extended from a busy channel to a faulty channel or a faulty switching element. Let the channel marked by an X be faulty. In CubeSR, a packet that needs to be transmitted from 0001 to 0000 will be blocked at 0001 creating a network failure. In CubeAR, however, the same packet can be transmitted through one of the alternate paths, for example, 0001 → 0011 → 0010 → 0000. This ability of CubeAR to route packets around faulty links makes CubeAR a fault-tolerant network. In fact, [ChS90] states and proves that the algorithm used in CubeAR, or algorithm A1 as referred to by [ChS90], can route messages between any two non-faulty nodes as long as the number of faulty components, links and nodes, is less than n, where n is the dimension of Cube. The results in this section extend the work of Chen and Shin by analyzing the algorithm’s performance through simulation and show the advantages of adaptive routing over static routing with respect to throughput, as well as fault-tolerance.

5.4 Summary

This chapter analyzed the packet-switched adaptive routed Cube. Three other works on Cubes with adaptive routing were outlined that all show the superiority of adaptive routing over static routing. The simulation results of this thesis showed that a Cube with adaptive routing (CubeAR) has lower steady-state delay than a Cube with static routing (CubeSR). CubeAR has a lower saturation point allowing higher throughput than CubeSR. Also, CubeAR can tolerate up to n-1 faulty components in the network, while CubeSR has no fault-tolerant capability. Based on these results, adaptive routing gives lower packet delay and higher throughput compared to static routing with an extra advantage of fault tolerance.
6. Conclusions

6.1 Summary of Research

A brief review of the material presented in this thesis is given before the conclusions and recommendations resulting from this study. In Chapter 1, an overview of restrictions and limitations of the traditional von Neumann computer was given. The research goals of this thesis were defined to focus the thesis on a few important factors.

Chapter 2 provided the background information necessary to understand the techniques used in parallel processing system evaluation. Different methodologies that classify types of parallel processing systems and an overview of interconnection networks were presented. The importance of a partitionable network in SIMD and MIMD machines was discussed.

Chapter 3 presented the methodology used in the modeling and the simulation of the multistage cube network (MSC) and the single-stage cube network (Cube). The SLAM II simulation language and the modeling assumptions used in the network simulations were introduced. The chapter also discussed how the networks are modeled using SLAM. Finally, a brief outline of the performance study of ASEN done by [RaR92] was given.

The simulation results of the partitioned MSC for a packet-switched environment were presented in Chapter 4. An analysis of the average time a packet takes to traverse input and output partitioned networks was presented. The comparison of the average packet delay of the MSC with partitioned stage and the MSC with randomly grouped PEs was given. The study of the partitioned MSC in a circuit-switched environment by [DaS85] was summarized.

Chapter 5 presented simulation results of the adaptive routed Cube. A brief presentation of other studies on Cubes with different adaptive routing schemes was given. The comparison of the average packet delay of an adaptive routed Cube and a static routed Cube was given.
6.2 Conclusions from Analysis

Several conclusions can be drawn from the simulation analysis done in this thesis. The ASEN outperformed the MSC for various network sizes, loading factors, source-destination distributions, and switching element sizes. The ASEN's higher performance came without an increase in the variance of packet delay and with lower implementation cost, i.e., fewer buffers were required. The performance benefits of the ASEN are attributed to the use of augmented links and priority queues.

When the performance difference between input and output partitions, two subtypes of stage partitioned MSC, was studied, the output partition performed only slightly better than the input partition such that it is reasonable to conclude that the two partitions performed equally. When these results were compared with the performance of the randomly grouped MSC, the partitioned MSC, both input partition and output partition, greatly outperformed the randomly grouped MSC. This result is supported by [DaS85] which showed the superiority of partitioned MSC over MSC with randomly grouped PEs in a circuit switching environment.

The simulation results showed that a Cube with adaptive routing has lower steady-state delay than a Cube with static routing. The adaptive routing has higher saturation point allowing higher throughput than the static routing. A Cube with adaptive routing has a fault-tolerant capability against faulty components in the network. Three other papers that showed higher throughput for adaptive routing when compared to static routing, outlined in Chapter 5, support the above results.

6.3 Recommendations for Future Research

A couple of proposed enhancements may form a base for future research in the area of interconnection network performance comparisons. The MSC analyzed in this thesis partitioned itself by setting the whole stages. A simulation model can be developed to implement an MSC which partitions itself by also setting a part of a stage both under packet and circuit switching.
methodologies. The adaptive routing algorithm used in this thesis can be implemented in a Cube with circuit switching to compare with the performance for packet switching. Implementing an average delay analysis of an adaptive routed Cube with injected faulty components will show throughput performance under faults.
References


Appendix A

Figure A.1 Performance of MSCPS and MSCRG
(1024 PEs Divided into 8 Groups with 128 PEs per Group)
Figure A.2 Performance of MSCPS and MSCRG
(1024 PEs Divided into 16 Groups with 64 PEs per Group)
Figure A.3 Performance of MSCPS and MSCRG
(1024 PEs Divided into 32 Groups with 32 PEs per Group)
Figure A.4 Performance of MSCPS and MSCRG
(1024 PEs Divided into 64 Groups with 16 PEs per Group)
Figure A.5 Performance of MSCPS and MSCRG
(64 PEs Divided into 4 Groups with 16 PEs per Group)
Figure A.6 Performance of MSCPS and MSCRG
(64 PEs Divided into 8 Groups with 8 PEs per Group)
Figure A.7 Performance of MSCPS and MSCRG
(64 PEs Divided into 16 Groups with 4 PEs per Group)
Appendix B

Figure B.1  Performance of Two Differently Routed Cubes  
(1024 PEs; Uniform Distribution)
Figure B.2 Performance of Two Differently Routed Cubes
(1024 PEs; Normal Distribution)
Figure B.3 Performance of Two Differently Routed Cubes
(64 PEs; Uniform Distribution)
Figure B.4 Performance of Two Differently Routed Cubes
(64 PEs; Normal Distribution)
Vita

Jahng S. Park was born on September 6, 1968 in Seoul, Korea. Because of his father who works for Korean Overseas Information Services and works overseas occasionally, he attended three years of elementary school in Hong Kong and attended high school in Fairfax County, Virginia. He completed his remaining elementary education and middle school in Seoul, Korea. For his last year of high school, he attended the Senior Experience Program at Thomas Jefferson High School for Science and Technology, a Virginia Governor's magnet school. He received his high school diploma from West Springfield High School, his base school. He then attended Virginia Tech majoring in electrical engineering. Upon receiving his Bachelor of Science degree in 1990, he continued to work towards his Master's degree at Virginia Tech. In August 1992, he applied for a leave of absence to fulfill the mandatory military service and returned to Korea. After returning to Virginia Tech in May 1993, he continued to work toward his Master's degree. He has been accepted into the Doctorate program by the Bradley Department of Electrical Engineering at Virginia Tech.

[Signature]