WASHINGTON UNIVERSITY
SEVER INSTITUTE OF TECHNOLOGY

ANALYSIS AND DESIGN OF HIGH-SPEED SWITCHING SYSTEMS FOR
ATM NETWORKS

by

NADER MIRFAKHRAEI

Prepared under the direction of Professor Jonathan S. Turner

A thesis presented to the Sever Institute of
Washington University in partial fulfillment
of the requirements for the degree of

DOCTOR OF SCIENCE

May, 1995

Saint Louis, Missouri
ABSTRACT

ANALYSIS AND DESIGN OF HIGH-SPEED SWITCHING SYSTEMS FOR
ATM NETWORKS

by Nader Mirfakhraei

ADVISOR: Professor Jonathan S. Turner

May, 1995
Saint Louis, Missouri

The integration of communication services in broadband networks and increasing traffic volume, require large-scale switching systems capable of handling high data rate operations. In this dissertation, we propose and analyze multipoint switching architectures aimed at high-speed ATM applications. We specifically emphasize the speed, performance, and scalability of the proposed switching systems, and strive to achieve these goals to support the needs of future communication networks.

We analyze a multicast ATM switching system using the Beneš topology with gigabit per second links. A complete evaluation is given at the switch level as well as the network level. Although various switches can, on paper, be designed to large dimensions, technological and physical constraints often impose a practical limit on their maximum size. The use of wafer-scale integration for switching systems studied here can potentially be a solution to this restriction. We propose and analyze a scalable multipath buffered crossbar (MBC) network with a fault-tolerant structure. We then extend the work on Manhattan-street networks (MSNs). MSNs belong to the class of deflection-routing networks. We present an ATM switch architecture using an MSN-based switch fabric, and through a comprehensive performance evaluation, study the efficiency of this architecture.

We describe routing and multicast algorithms for the switching architectures presented in this thesis and develop analytic models to evaluate their queueing performance. Finally, we develop simulations for traffic evaluation of the MSN. Comparisons of the results of simulation and analysis show that analysis provides acceptable accuracy.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>Dissertation Objectives and Overview</td>
<td>3</td>
</tr>
<tr>
<td>Dissertation Overview</td>
<td>4</td>
</tr>
<tr>
<td>2 REVIEW OF BROADBAND ATM SWITCHING SYSTEMS</td>
<td>7</td>
</tr>
<tr>
<td>ATM Switching Systems</td>
<td>7</td>
</tr>
<tr>
<td>Classification of ATM Switching Systems</td>
<td>8</td>
</tr>
<tr>
<td>Switching Network Topologies</td>
<td>9</td>
</tr>
<tr>
<td>Characteristics of ATM Switching Systems</td>
<td>10</td>
</tr>
<tr>
<td>Techniques for Improving Performance</td>
<td>12</td>
</tr>
<tr>
<td>Remarks</td>
<td>13</td>
</tr>
<tr>
<td>3 A GIGABIT MULTISTAGE ATM SWITCHING SYSTEM</td>
<td>14</td>
</tr>
<tr>
<td>Architecture of the Network</td>
<td>14</td>
</tr>
<tr>
<td>Routing and Multicast Algorithms</td>
<td>16</td>
</tr>
<tr>
<td>An Example for Multicasting Cells</td>
<td>18</td>
</tr>
<tr>
<td>Overview of the Switch Element</td>
<td>19</td>
</tr>
<tr>
<td>Cell Format</td>
<td>20</td>
</tr>
<tr>
<td>Data Slices</td>
<td>22</td>
</tr>
</tbody>
</table>
Controller Unit .................................................. 23
Complexity of Switch Element ................................. 25
Main Switch Functions ........................................... 25
Buffer Control Mechanism ...................................... 26
Contention Resolution Process and Cell Replication ........ 27
Flow Control ....................................................... 30
Hardware Simulations Results ................................. 31
Performance Evaluation .......................................... 33
Evaluation of Complexity ........................................ 40
Remarks ............................................................. 41

4 EVALUATION OF WSI FOR SWITCHING SYSTEMS .......... 42
Wafer-Scale Integration .......................................... 43
Defect Tolerance and System Reconfiguration ............... 44
Yield Analysis ...................................................... 45
Cost of Interconnections in Multistage Switching Networks 47
A Modified Version of Multi-Stage Network for WSI ........ 49
Network Controller and Delay Issue ............................ 50
Evaluation of Complexity ........................................ 53
Remarks ............................................................. 54
5 A MULTIPATH BUFFERED CROSSBAR FOR WSI

System Architecture ........................................... 55
Design of a Multipath Buffered Crossbar .................... 56
Input and Output Port Processors ............................ 59
Structure of a Self-Driven Crosspoint ....................... 61
Defect Tolerance and WSI Reconfiguration ................... 63
Performance Evaluation ........................................ 63
Queueing Model ................................................ 64
Throughput and Delay .......................................... 70
Performance Evaluation Results .............................. 71
Effect of Increasing Offered Load ............................ 71
Effect of Increasing Number of Ports ....................... 74
Effect of Increasing Number of Columns .................... 77
Evaluation of Complexity ..................................... 77
Estimation of Area ............................................. 78
Evaluation of Yield for WSI Technology ..................... 79
Remarks ......................................................... 80

6 A MANHATTAN STREET SWITCHING NETWORK ............ 81

Manhattan Street Network .................................. 82
Routing in the MSN ........................................... 83
Node Structure with Shared Buffering ....................... 85
Description of Node Operation ........................................ 87
Port Processors ............................................................ 88
Number of Nodes of Specific Type ................................... 88
Performance Analysis ..................................................... 90
Example of Multicast Implementation ............................... 101
Evaluation of Complexity .............................................. 104
Estimation of Area ........................................................ 105
Evaluation of Yield for WSI Technology .......................... 106
Defect Tolerance ......................................................... 107
Remarks .................................................................. 108

7 SIMULATION OF MANHATTAN STREET NETWORK .......... 109
Simulation Structure ..................................................... 109
Cell Arrival ............................................................... 110
Routing Strategy ......................................................... 111
Contention and Deflection ............................................. 111
Traffic Flow .............................................................. 112
Simulation Results and Comparison with Analysis ............. 112
Collected Data on the Number of Flowing Cells ............... 113
Effect of Increasing Offered Traffic Load .......................... 117
Effect of Network Size .................................................. 119
Other Results of Simulation ........................................... 121
Effect of Large Shared Buffers ........................................ 121
Effect of the Speed Advantage ........................................ 124

8 CONCLUSIONS ......................................................... 126

Summary and Contributions ........................................... 126
Comparison of Wafer-Scale Switches ................................. 128
Extension and Future Work ........................................... 129

ACKNOWLEDGMENTS .................................................. 132

ACRONYMS AND SYMBOLS .......................................... 134

Acronyms .............................................................. 134
Symbols .............................................................. 136

BIBLIOGRAPHY ....................................................... 137

VITA ........................................................................ 145
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>An ATM cell and its header structure.</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>The overview of a typical ATM switching system.</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>Classification of switching network topologies.</td>
<td>9</td>
</tr>
<tr>
<td>2.4</td>
<td>Examples of single-path switching networks, (a) a crossbar switch as a</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>single-stage network, (b) a delta network as a multistage network.</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>Clos network, as an example of a multipath/multistage switch.</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>An overview of the switching network.</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>Multicast algorithm for the copy network.</td>
<td>17</td>
</tr>
<tr>
<td>3.3</td>
<td>Cell routing algorithm for the routing network.</td>
<td>18</td>
</tr>
<tr>
<td>3.4</td>
<td>A multicast routing example with a global fanout of 5.</td>
<td>19</td>
</tr>
<tr>
<td>3.5</td>
<td>Bit-sliced structure of the switch element.</td>
<td>20</td>
</tr>
<tr>
<td>3.6</td>
<td>Cell format for the gigabit switching system.</td>
<td>21</td>
</tr>
<tr>
<td>3.7</td>
<td>Data slice of the switch element.</td>
<td>22</td>
</tr>
<tr>
<td>3.8</td>
<td>The switch element controller.</td>
<td>24</td>
</tr>
<tr>
<td>3.9</td>
<td>Functional diagram of the <em>buffer control circuit</em> (BCC).</td>
<td>27</td>
</tr>
<tr>
<td>3.10</td>
<td>A column of <em>arbitration elements</em> (AEs), and the contention bus structure.</td>
<td>28</td>
</tr>
<tr>
<td>3.11</td>
<td>Functional diagram of the <em>upstream grant circuit</em> (UGC).</td>
<td>30</td>
</tr>
</tbody>
</table>
3.12 Timing simulation of the test chip for two contending cells. The comparison of two internal signals $a_{03.0}$ and $a_{03.1}$ starting from clock cycle 20: the cell with the higher priority ($\text{req.priority} = 0001100-000101$) wins the contention. .................................................. 32

3.13 Throughput ($T$) vs offered load ($\rho$) when the switch size $d = 4$ and number of shared buffers per port of switch ($B$) varies. ......................... 38

3.14 Throughput ($T$) vs offered load ($\rho$) when the switch size $d = 16$ and number of shared buffers per port of switch ($B$) varies. ......................... 38

3.15 The effect of enlarging network size ($n$) on throughput ($T$) for two different switch sizes ($d$) where the number of buffers per port of switch $B = 2$. .................................................. 39

3.16 The effect of enlarging network size ($n$) on throughput ($T$) for two different switch sizes ($d$) where the number of buffers per port of switch $B = 3$. .................................................. 39

3.17 Complexity of the network per port (in million transistors) vs the network size ($n$) and the number of buffers per port ($B$). ........................... 40

4.1 Method of bypassing defective integrated circuit modules in WSI. ....... 45

4.2 Realization of the wiring in the first stage of a Beneš network. ......... 48

4.3 A modified network architecture using shared buffer crossbars. ....... 50

4.4 The copy network using a bit-slice shared buffer crossbar structure. ... 51

4.5 The uniform structure of the network data slice designed for WSI. .... 52

4.6 Complexity of the network per port (in million transistors) vs the network size ($n$) and the number of buffers per port ($B$). ........................... 53
5.1 Overall architecture of the switching system using WSI technology. 56
5.2 Routing and multicasting in the multipath buffered crossbar \((n = k = 4)\). 57
5.3 Multicast connections and the port processors. 60
5.4 The structure of a \textit{self-driven crosspoint} for the proposed MBC network. 62
5.5 A queueing model for the multipath buffered crossbar. 64
5.6 Signaling in the multipath buffered crossbar. 67
5.7 Markov chain model of the input port queue. 68
5.8 Throughput \((T)\) of a 16-port network when offered load \((\rho)\) and number of columns \((k)\) vary. 72
5.9 Delay \((D)\) in a 16-port network when offered load \((\rho)\) and number of columns \((k)\) vary. 72
5.10 Throughput \((T)\) of a 64-port network when offered load \((\rho)\) and number of columns \((k)\) vary. 73
5.11 Delay \((D)\) in a 64-port network when offered load \((\rho)\) and number of columns \((k)\) vary. 73
5.12 Throughput \((T)\) when the network dimension \((n)\) and the number of columns \((k)\) simultaneously increase. 75
5.13 Delay \((D)\) when the network dimension \((n)\) and the number of columns \((k)\) simultaneously increase. 75
5.14 Throughput \((T)\) versus the number of columns \((k)\) where the number of buffers per crosspoint \((\beta)\) is 1 or 2. 76
5.15 Average network delay \((D)\) versus the number of columns \((k)\) where the number of buffers per crosspoint \((\beta)\) is 1 or 2. 76
5.16 Complexity of the network per port (in millions of transistors) vs the network size \((n)\) and the number of buffers at the crosspoint \((\beta)\). 77

5.17 Evaluation of the network area on a single wafer versus the technology feature size \((t)\). 79

6.1 A 4×4 Manhattan-street network. 82

6.2 Preferred directions and relative addresses. 83

6.3 The structure of a node. 86

6.4 Distribution of traffic load in a node. 95

6.5 Throughput \((T)\) versus offered load \((\rho)\) when the number of shared buffers per port of node \((B)\) varies. 100

6.6 Delay \((D)\) vs offered load \((\rho)\) when the number of shared buffers per port of node \((B)\) varies. 100

6.7 A multicast tree. 102

6.8 The routing example of a cell with a fanout 4 in the Manhattan street network. 103

6.9 Complexity of the network per port (in millions of transistors) vs the network size \((n)\), and the number of shared buffers per port of node \((B)\). 105

6.10 Evaluation of the network area on a single wafer versus the technology feature size \((t)\). 106

7.1 Modeling a node for queueing-deflection analysis. 110

7.2 Throughput \((T)\) versus offered load \((\rho)\) when the number of shared buffers per port of node \((B)\) varies. 118
7.3 Delay \((D)\) vs offered load \((\rho)\) when the number of shared buffers per port of node \((B)\) varies. ......................................................... 118

7.4 The effect of enlarging network size \((n)\) on throughput \((T)\) for three different numbers of buffers per port of node \((B)\) when the speed advantage is one. ......................................................... 120

7.5 The effect of enlarging network size \((n)\) on throughput \((T)\) for three different numbers of buffers per port of node \((B)\) when the speed advantage is \(3N/4\) \((n = N^2)\). ......................................................... 120

7.6 The effect of increasing the number of shared buffers per port of node \((B)\) on throughput \((T)\) when offered load \((\rho)\) varies. ................................. 122

7.7 The effect of increasing the number of shared buffers per port of node \((B)\) on delay \((D)\) when offered load \((\rho)\) varies. ................................. 122

7.8 The effect of increasing the number of shared buffers per port of node \((B)\) up to \(10\) on the average number of deflections \((\Delta)\) for three different offered loads \((\rho)\). ......................................................... 123

7.9 The effect of speed advantage \((s)\) on the throughput \((T)\) of the network when offered load \((\rho)\) varies \((n = 16)\). ......................................................... 124

7.10 The effect of speed advantage \((s)\) on the throughput \((T)\) of the network when offered load \((\rho)\) varies \((n = 16)\). ......................................................... 125
LIST OF TABLES

Table | Page
---|---
5.1 Yield \((y)\) evaluation for the crosspoint and the port processors of the multipath buffered crossbar. | 80
6.1 Examples on the number of don't-care nodes for different network sizes. | 90
6.2 The exact values for the number of additional hops \((e)\) a cell takes as a result of the fact that streets are always one way in all regions of the MSN. | 97
6.3 The average number of deflections \((\Delta)\) versus offered load \((\rho)\) when the number of shared buffers per port of node \((B)\) vary. | 101
6.4 Yield \((y)\) evaluation for the node of the Manhattan street network. | 107
7.1 Collected data from the simulation of a 16-node network: the number of cells of type 2, 3, and 4 from quadrant \(Q_1\). | 114
7.2 Collected data from the simulation of a 16-node network: the number of cells of type 2, 3, and 4 from quadrant \(Q_2\). | 114
7.3 Collected data from the simulation of a 16-node network: the number of cells of type 2, 3, and 4 from quadrant \(Q_3\). | 115
7.4 Collected data from the simulation of a 16-node network: the number of cells of type 2, 3, and 4 from quadrant \(Q_4\). | 115
7.5 Collected data from the simulation of a 64-node network: the number of cells of type 2, 3, and 4 from quadrant \(Q_1\). | 116
7.6 Comparison of simulation and analysis on the overall percentage of “type-4” cells (in the analysis we use the equivalent term of “don’t-care nodes”). 116

7.7 Comparison of simulation and analysis on the average number of deflections (Δ) versus offered load (ρ) when the number of shared buffers per port of node (B) vary. .............................. 119
ANALYSIS AND DESIGN OF HIGH-SPEED SWITCHING SYSTEMS FOR ATM NETWORKS

1. INTRODUCTION

We are witnessing an explosive demand for communications from homes, businesses, cities, states and countries. The design of modern communication systems is motivated by the desire to meet this demand and to integrate capabilities of networks all in a Broadband Integrated Services Digital Network (B-ISDN). Advances in technology allow for the merging of telephone networks and computer networks that were once separate and distinct. The B-ISDN is required to support the exchange of multiple types of information such as voice, video, and data among multiple types of users, while satisfying the performance requirement of each individual application. The expanding diversity of high-bandwidth communication applications, consequently, calls for a unified, flexible, and efficient network.

In the past, networks were developed for new services. As the number and variety of services grow, this is no longer a cost-effective way to meet emerging communication needs. In response to the expected increase in services, Asynchronous Transfer Mode (ATM) was selected\(^1\) to be the standard of communications to support the integration of diverse traffic types over the B-ISDN. ATM is an evolving transfer mode receiving

\(^1\)by CCITT (International Consultative Committee for Telecommunications and Telegraphy)
much attention in the literature and industry. The selection of ATM, in fact, responds to the growing demand for higher bandwidth communication networks that interconnect a large number of users. The ATM technology allows the integration of services since the user interfaces can operate independently from the traffic source characteristics.

An essential component of the B-ISDN is the switching network. The increase in traffic volume calls for large, high-speed, and high-performance switching systems. A switch must deal with output contention, where cells arriving at the same time on inputs of the switch may request a common output. Under such conditions traffic congestion may occur. Even if a network is naturally nonblocking, the switch fabric is still unable to provide services for more than one cell simultaneously requesting the same output. There are several feasible solutions to this problem which will be explained in the later chapters. One common approach, for example, is to utilize buffers within the switch. In a buffered switch, if two or more cells request a switch output at the same time, one cell is allowed to leave the switch while the others remain in the buffer for the next cycle rather than being discarded.

The design of modern switching systems must also meet the requirements for new communications applications. A switching system, in this context, must support multirate connections, unlike the classical switches designed for single-rate services. The capability of multicasting traffic is another fundamental factor. ATM switching systems must be able to multicast a message to many users and even route many messages to one user. In the multipoint environment, traditional networks are subject to severe blocking since they are not typically designed for high bandwidth communications. Under both conditions, multirate and multipoint, the performance of switching networks closely relies on the network architecture. The efficiency also depends on how fast networks can transfer the traffic.
The inclusion of all of the above features (like buffering and multicasting capability) into modern switching systems requires considerable sophistication in the architecture. Simplistic approaches can fail to meet required performance or cost objectives.

1.1. Dissertation Objectives and Overview

The objectives of this research are summarized as follows:

There are two distinct factors contributing to system performance: data path bandwidth and queueing delay. The data path bandwidth is determined by clock speed and data path width. The given circuit technology places a limit on the practical clock rates that can be achieved. Further increases can be obtained only by increasing the data path width, which directly affects the interconnection complexity of the switching network. In Chapter 3, we will explore the implications this has on the design of a switching element suitable for the construction of a gigabit switching system. In Chapter 4 we will introduce the concept of wafer-scale integration (WSI) and explore its potential for reducing the constraints introduced by growing interconnection complexity. The potential advantages of WSI include: 1) increased speed and reliability, 2) reduced power, 3) reduced transmission line reflection of signals, and 4) reduced area, weight and volume. Such potential advantages are mainly attributable to shorter interconnections, removal of most of the output pad drivers, and removal of individual packaging. As interconnection complexity grows with network size and data rate, these advantages take on added significance. Because WSI technology alters underlying design trade-offs, it's important to re-evaluate switch architectures in light of the changing trade-offs. In the later chapters of this dissertation we consider architectural approaches that seem to be particularly suitable to the wafer scale environment.
For the network architectures proposed in this thesis, we evaluate the queueing performance through analytic approaches and, where appropriate, simulations. We present analytical queueing models and use them to determine system throughput and delay as a function of offered traffic load, number of internal buffers, and network size.

1.1.1. Dissertation Overview

This dissertation proposes, analyzes, and compares different high-speed switching architectures for ATM communications applications and is organized as follows:

Chapter 2 gives an overview of typical modern switching systems. In this chapter, we also present a short survey of ATM switching systems and summarize the methods by which system performance can be enhanced.

Chapter 3 presents the implementation and analysis of a multistage ATM switching system for use in a multicast ATM network supporting gigabit per second links. In this chapter we present a complete work at both the network and switch level to obtain insight into design aspects and to use this insight as the basis for the switch architectures presented in the next chapters. The work includes architectural design, multicast and routing algorithms, switch-level design, switch-level timing simulation, traffic performance analysis, and complexity evaluation. The system is constructed using the Beneš topology, and the switch elements consist of identical parallel data slices and a global controller. Each data slice uses shared buffers. The controller determines which buffered cells are to be sent to the outputs based on an internal contention resolution process and generates grant flow control through the network to avoid the overflow of buffers. The performance evaluation is also given in this chapter to estimate the throughput in response to different offered traffic loads and network sizes. Some timing simulations derived from the circuit layout will be presented to evaluate the implementation of the
switch controller. We have actually designed switch element prototype test-chips using CMOS VLSI technology and tested the fabricated chips at a clock speed of 100 MHz.

Chapter 4 evaluates the use of wafer-scale integration (WSI) technology for switching systems and addresses related design issues. We propose the use of wafer-scale integration technology, which allows the entire system to be implemented on a single semiconductor wafer. WSI permits us to consider new designs which would not be practical in a system using VLSI technology. The evaluation study demonstrates that multistage switching networks which are typically the best choice for VLSI technology are not the best choice when using WSI. Also, in this chapter a modified version of the multistage switching network introduced in Chapter 3 is presented.

Chapter 5 extends the discussion in Chapter 4 and presents the design and analysis of a new switching system architecture which is appropriate for wafer-scale implementation. We intend this architecture for high-speed, multipoint ATM applications. The system is composed of a large, multipath, buffered crossbar with an expandable structure which can easily be scaled up or down according to the choice of wafer size. The crossbar consists of buffered crosspoints each with two switches. With this configuration, an incoming cell needs to pass through only two stages of queueing to reach its output port. In this chapter we present a queueing model and discuss the performance evaluation of the network with analysis. We also assess the proposed system from the complexity, yield, and area points of view for the use of wafer-scale integration technology.

Chapter 6 considers the Manhattan-street network (MSN) as another alternative for implementing an ATM switching system, suitable for wafer-scale implementation. The MSN belongs to a different class of networks in which cell contention resolution is based on deflection. We develop an analytical queueing model for an MSN in which each node has a shared buffer and we use this model to evaluate its performance. We
study the system throughput and delay versus different variables including the offered load, network size, and number of buffers. We finally investigate the compatibility of such a network with WSI technology and study some different integrated circuit aspects such as complexity, yield, and area.

Chapter 7 evaluates the performance of the Manhattan street network described in Chapter 6 using simulation. The simulation is developed for this switching system to verify the traffic performance analysis. We evaluate the effects of offered traffic loads, large shared buffers, speed advantage, and network size on the network performance. We provide various statistical results which are used to justify various approximations in the analysis.
2. REVIEW OF BROADBAND ATM SWITCHING SYSTEMS

In this chapter we review the basic concepts of ATM switching systems. We also classify the network topologies suitable for ATM applications and discuss the characteristics of practical ATM architectures.

2.1. ATM Switching Systems

Asynchronous transfer mode (ATM) was modeled on the idea of a worldwide network, and it may well be the technology that brings the communications and computer industries together. The objective of ATM technology is to provide a homogeneous backbone network in which all types of traffic are transported with the same small fixed sized cells. One of the key advantages of ATM systems is flexible multiplexing to support multiple forms of data. The sources that ATM is expected to support can be bursty, such as fax, coded video and bulk data transfer.

![Figure 2.1: An ATM cell and its header structure.](image)

Regardless of traffic types and the speed of sources, the traffic is converted into 53-byte ATM cells as shown in Figure 2.1. Each cell has a 48-byte data payload and a 5-byte header. The header identifies the virtual channel to which the cell belongs. The header consists of several fields including a two-level addressing structure of virtual path identifier (VPI) and virtual circuit identifier (VCI). Together, the VPI and VCI provide
two levels of multiplexing. The other fields are *generic flow control* (GFC), *payload type* (PT), *cell loss priority bit* (CLP), and *header error control* (HEC). Details of the ATM standards can be found in [48].

In Figure 2.2, an abstract model of an ATM switching system is shown. Cells arrive at \( n \) input ports and are routed out from \( n \) output ports. The system consists of three main parts, *port processors* at both the input and output sides, and a *switching network*. When a cell carrying VCI \( b \) arrives from a given link \( i \), its VCI is used to index a *virtual circuit translation table* (VXT) in the corresponding input port processor to identify the output link address \( j \) and a new VCI \( c \). In the switching network, cells are routed to the desired outputs. As shown in Figure 2.2, a cell can also be multicast to more than one output. Finally, in the output port processors, cells are buffered, and in some switch architectures cells are resequenced in order to avoid misordering. In addition to these duties, there are a number of other important processes and functions taking place in each of the mentioned blocks which we will explain in detail in the next chapters.

### 2.2. Classification of ATM Switching Systems

There have been a number of papers in the literature such as [16, 19, 47, 52, 59, 62, 64] presenting architectures for ATM switching systems. There are also a few papers [1, 46,
55] that compare the switching systems from different perspectives, providing a comprehensive survey of different systems. Many different factors can be used to characterize switching systems, including buffering, capability of multipoint connections, speed, performance, cost, reliability, fault tolerance, and scalability. In the following sections we focus on the topology of the interconnection networks that form the heart of modern switching systems and compare them from various standpoints.

2.2.1. Switching Network Topologies

The existing switching networks can be categorized into two major groups: single-path and multipath networks as shown in Figure 2.3. In a single-path network, between each input port and output port there exists exactly one route. This property can, however, be a source for traffic congestion and traffic blocking. In a multipath network, any connection can be established through more than one path. We can further classify each of these two groups of networks into single-stage and multistage networks. In a single-stage network a connection is established through one stage of switching where
in a multistage network a cell must pass through a number of switching stages. The number of stages in a multistage network often grows with increasing network size.

A basic single-path/single-stage structure is a crossbar where any connection is made through one switch as shown in Figure 2.4 (a). In this group, other architectures can be mentioned, such as the Knockout switch [64], the bus switch [18], and the ring switch [54]. Figure 2.4 (b) shows a delta network as an example of the single-path/multistage networks. Multistage networks can provide lower complexity than single stage networks, while providing comparable performance. There are a few other networks in this group, such as the banyan and omega networks.

The multipath buffered crossbar (MBC) [38] which will be presented in Chapter 5 is an example of a multipath/single-stage architecture. In the MBC any connection can be established through many different paths but always requiring only one stage consisting of two crosspoints (see Figure 5.2). There are several networks in the category of multipath/multistage networks such as the Clos network [14], the Batcher sorting network [3], the Beneš network [4], the Cantor network [9], and the extended delta network. Figure 2.5 shows the Clos network as an example for this group.

2.2.2. Characteristics of ATM Switching Systems

At this point, we further extend the discussion and itemize the features and options that can be used in ATM switching systems:

- **Buffered versus Unbuffered.** One feasible solution to reduce traffic congestion is to use buffers in each switch element of a network. One of the original analyses of networks with buffering in the literature was introduced by Dias [17]. In this paper, the use of buffers in delta networks is analyzed. This idea was further developed later for various switching systems.
Figure 2.4: Examples of single-path switching networks, (a) a crossbar switch as a single-stage network, (b) a delta network as a multistage network.

Figure 2.5: Clos network, as an example of a multipath/multistage switch.

- **Use of Flow Control.** Flow control can be provided between stages in a multistage network to regulate the flow of cells and prevent buffer overflow. The provision of flow control does require some extra circuitry, but this is usually more than offset by the reduction in buffer capacity it allows.

- **Discard versus Deflection.** At each switching stage of a network, a conflict may arise since a number of cells may request the same output. In networks without flow control, arriving cells that cannot be buffered can either be *discarded* (to be
described in Chapter 3) or deflected [16] [32]. (the deflection routing technique will be used in Chapter 6). We can also combine the above two methods.

- **Multicast and Broadcast Feature.** One of the features that modern switching networks are expected to have is the capability of multicasting (copy to any subset of the outputs). The broadcast packet switching network proposed in [59] is one of the original system architectures that supports multicast communications. Also, Turner recently proposed a new efficient, *cell recycling technique* [62] for multicast connections. Using this method cells are recycled through the switch fabric and are duplicated on each successive pass.

2.3. Techniques for Improving Performance

There are several practical methods to increase throughput and reduce the possibility of blocking as follows:

- **Buffering.** We explained earlier that the use of buffers in switching networks reduces the traffic congestion and thus increases the system throughput. By utilizing buffers in a switch, cells are not discarded when they request the same switch output but instead are kept in buffers for a later contention resolution process.

- **Combined Networks.** Combined networks basically provide extra stages and produce extra paths, although a combined network can also be designed for other purposes such as multicasting. Examples include the Sunshine switch [19], the Starlite switch [25], Lee's switch [29], and Turner's switch [59].

- **Randomizing Traffic.** This technique is used to distribute traffic evenly across the switching network to prevent local congestion.
• **Recirculation of Traffic.** When a cell is not successfully delivered to a given port, it can be recirculated. The recirculated cells will contend for delivery (possibly with higher priorities) in the next cycle. Two examples of architectures that use this technique are the Starlite switch [25] and Sunshine switch [19].

• **Speed Advantage.** Speed advantage in a switching system refers to the ratio of the internal link speed to the external link speed (to be used in Chapter 7).

2.4. Remarks

In this chapter, we reviewed the basic concepts in ATM switching systems and summarized the characteristics of such systems. We also introduced some of the previous work on this subject which will form the basis for extensions to be introduced in later chapters.
3. A GIGABIT MULTISTAGE ATM SWITCHING SYSTEM

This chapter\textsuperscript{1} presents the structure and design of a high-speed ATM switch supporting link speeds in the range of gigabits per second. The internal structure of this system will be used as a framework to develop the switch architectures of the next chapters. This system is able to support point-to-point connections as well as multipoint connections. The switch elements use a bit-sliced architecture and contain internal buffering. The switch elements also generate flow control signals in the network to avoid the overflow of buffers.

This chapter is organized as follows. In Section 3.1, the architecture of the network is introduced. Section 3.2 presents multicast and routing algorithms. Section 3.3 analyzes the switch element structure which includes the parallel architecture of the data part and the switch controller. An overview of the implementation of cell replication will also be presented in this section. The details of the switch design, and the discussion of the contention resolution process, multicasting implementation, and flow control mechanism appear in Section 3.4. Chip simulation results are discussed in Section 3.5, and the complexity of the system is evaluated in Section 3.7.

3.1. Architecture of the Network

The system is designed to handle a clock speed of 100 MHz, and it uses byte wide data paths. The network is able to support point-to-point connections as well as multipoint connections. An overview of the switching system is shown in Figure 3.1. Incoming cells enter the network from \textit{n} fiber optic links (FOL). \textit{Input port processors} (IPPs)

\textsuperscript{1}This chapter has been published in part in [35], [37], [41], [42].
convert cells into bit-sliced form. The details of the bit-sliced format will be explained in Section 3.3. In the IPP, routing information and the global fanout are also attached to a cell. The global fanout refers to the total requested number of the copies of a cell for multipoint connections.

![Diagram]

**Figure 3.1:** An overview of the switching network.

Each switch element within the switch fabric is constructed with \( m \) identical data slices and a controller. The routing information is processed within the controller. Switch elements use internal queuing in the form of shared buffers. In order to reduce cell loss and enhance the network performance, the number of internal buffers (\( b \)) is selected to be larger than the number of inputs to the switch element (\( d \)). The multicast implementation of the network is similar to the one explained in [59] where a copy network (CN) replicates cells as specified by the control information. The copy network uses the Beneš topology [4]. Using this network with \( n \) input/output ports constructed with \( d \times d \) switches (\( B_{n,d} \)), the total number of switch elements (\( s \)) is equal to:

\[
s = \frac{n}{d} \cdot (2[\log_d n] - 1) \tag{3.1}
\]

where \( 2[\log_d n] - 1 \) is the total number of stages and \( n/d \) is the number of switch elements per stage. Multiple, identical copies of the cells arrive at the broadcast translation circuits (BTCs) where the routing information of each cell is altered. The BTCs
have processors which compare the broadcast channel numbers (BCNs) of the cells with their own data to identify proper switch fabric outputs. In the case of point-to-point connections, the BTCs do not change the routing information of the header. In the routing network (RN), all copies of the cells get routed based on the routing instructions provided by the BTCs and are guided to output port processors (OPPs). In each OPP, a cell is resequenced and buffered. A resequencing operation is needed for misordered cells. This misordering is usually caused by different paths and different delays in the queues through the network. In addition, the OPPs reconvert cells from the bit-sliced to the original form and send them to the external fiber optic links (FOLs).

### 3.2. Routing and Multicast Algorithms

Figures 3.2 and 3.3 present algorithms for multicasting and routing cells within the copy network and routing network, respectively. These algorithms are implemented in the controller of the switch element. The main variables for the algorithms are defined as follows:

- \( n \): network dimension;
- \( d \): switch element dimension;
- \( k \): \([\log_d n]\);
- \( j \): the stage number (left to right), \( j \in \{1, 2, ..., 2k - 1\} \);
- \( r \): \(2k - 1 - j\);
- \( F \): global fanout, \( F \in \{0, 1, ..., n - 1\} \);
- \( F_j \): fanout remaining when cell arrives at stage \( j \);
- \( f_j \): local fanout at stage \( j \);
- \( g_j \): group size;
- \( u_j \): group index;
begin
  for $1 \leq j \leq k - 1 \implies$
  \[ F_j = F; \]
  \[ f_j = 1; \]
  rof

  for $k \leq j \leq 2k - 1 \implies$
  \[ r = 2k - 1 - j; \]
  \[ F_j = \left\lfloor \frac{F_{j-1}}{f_j} \right\rfloor; \]
  \[ f_j = \left\lfloor \frac{F_j}{d} \right\rfloor; \]
  \[ g_j = \left\lfloor f_j \right\rfloor^{(2)} \text{ the smallest power of 2 greater than or equal to } f_j; \]
  \[ u_j = \text{random number } \in [0, (d/g_j) - 1]; \]
  \[ v_j = (2^{g_j}u_j + f_j - 1) - (2^{g_j}u_j - 1); \]
  rof
end

Figure 3.2: Multicast algorithm for the copy network.

$v_j$: bit vector which specifies outputs that receive copies;

$A$: output address of cell;

$a_j$: address in stage $j$.

In the copy network, cells are replicated as designated by the initial global fanout ($F$) given in the routing control field of the cell header. The copying method is such that the replication of the cells takes place stage by stage within the network in order to distribute the traffic as evenly as possible. There are two types of routing: point-to-point and multipoint connections. In the case of a point-to-point connection, the cell is routed randomly in the first $k - 1$ stages and routed according to successive $d$-array digits of the destination address in the last $k$ stages. When multipoint connections are requested, the cell is routed randomly in the first $k - 1$ stages, and in the last $k$ stages
\begin{verbatim}
begin
  for 1 \leq j \leq k - 1 \rightarrow
    a_j = \text{random number} \in [0, d-1];
  rof
  for k \leq j \leq 2k - 1 \rightarrow
    a_j = \left\lfloor \frac{j}{d^j} \right\rfloor \mod(d);
  \nu_j = 2^\nu_j;
rof
end
\end{verbatim}

Figure 3.3: Cell routing algorithm for the routing network.

the cell is copied. However, it may sometimes occur that the final number of copies that
appear on the outputs is slightly more than requested, in which case unnecessary copies
of cells can be easily thrown away. This technique reduces the hardware complexity in
the controller.

In the routing network, cells are only guided to the requested outputs. In [42] a
similar algorithm is presented in which the number of copies is always a power of two.

3.2.1. An Example for Multicasting Cells

To make the algorithms clearer, follow the example in Figure 3.4 which shows a network
with \( n = 16 \) ports using switch size \( d = 4 \). Assume the global fanout of an incoming
cell is \( F = 5 \) and that copies of the cell are to be routed to output port numbers 1, 6,
9, 12, and 14. In the copy network, there are \( (2k - 1) = 3 \) stages \( (k = \lfloor \log_d n \rfloor = 2) \);
at stage \( j = 1 \), \( F_1 = 5 \) and the local fanout \( f_1 = 1 \), so the cell gets distributed. At
stage \( j = 2 \), \( F_2 = \left\lfloor F_1/f_1 \right\rfloor = 5 \) and \( f_2 = \left\lfloor F_2/d^r \right\rfloor = 2 \), thus, two copies are made at
this stage and guided to two switches at the last stage \( (j = 3) \). At the third stage,
\( F_3 = \left\lfloor F_2/f_2 \right\rfloor = 3 \) and the local fanout \( f_3 = \left\lfloor F_3/d^r \right\rfloor = 3 \), therefore, for each of these
switches, three copies of the cell are made. Notice, the sum of the above copies (6) has exceeded the requested global fanout (5), so as mentioned, the one additional copy is thrown away.

3.3. Overview of the Switch Element

As explained briefly above, the bit-sliced technique is utilized in the structure of the switch elements. In the bit-sliced form as shown in Figure 3.5, a cell is converted to \( m \) bit slices. Any switch element also comprises \( m \) identical data slices labeled DS(0) through DS(\( m - 1 \)) and a controller. In a \( d \times d \) switch, cells enter on one of the \( d \) upstream data lines (inputs), ud(0) through ud(\( d - 1 \)), so that \( m \) bit slices of each cell are distributed in parallel across \( m \) separate data slices. Cells exit from the switch element on the downstream data lines (outputs), dd(0) through dd(\( d - 1 \)). Each data slice contains \( b \) shared buffers to store several cells for each output port. The controller shown at the bottom of Figure 3.5 controls the operation of the switch element. A set of \( d \) downstream grant signals, dg(0) through dg(\( d - 1 \)), which function as flow controls, report the situation of buffers in downstream neighbors to the controller. When the controller completes a cell operation, and is ready for new requests on its upstream data lines, it generates and sends a corresponding set of \( d \) upstream grant signals,
ug(0) through ug(d – 1), to the upstream neighbors. In order to monitor the control information for all cells entering the switch, an individual data slice labeled DS(0) is used to buffer the first stream of a cell containing control information. This data together with downstream grants control the flow of cells into and out of the shared buffer.

3.3.1. Cell Format

The cell comprises a data part using a standard ATM cell (8 × 53 bits) and a header (control part) as shown in Figure 3.6. The control part contains the routing and fanout information consisting of four fields: a routing control indicating whether the cell should be copied or not, a data/control (D/C) flag specifying the presence of a cell if it is 1, the routing field comprising the destination address or the fanout of a multipoint cell, and finally, the Time field which contains a timestamp used for cell resynchronization. Also
Figure 3.6: Cell format for the gigabit switching system.
a parity bit is added to both the data and control parts for error detection purposes. Between each two cells, one bit is considered idle as a guard band.

3.3.2. Data Slices

The implementation of the data slice is illustrated in Figure 3.7. Any data slice has $d$ upstream data lines, $ud(0)$ through $ud(d-1)$, as inputs and $d$ downstream data lines, $dd(0)$ through $dd(d-1)$, as outputs. Every data slice contains $b$ one-slot buffers, $BS(0)$ through $BS(b-1)$; each can keep one bit slice of a cell ($55 \times 1$ bits). A cell entering the switch element is directed to one available buffer slot via a $d \times b$ input crossbar. After the completion of the contention process, a cell and its copies exit the switch through a $b \times d$ output crossbar. The switching operations of the crosspoints at the input and output crossbars are implemented by their own local signals generated by input crossbar control and output crossbar control. These two parts are in turn controlled by the main controller of the switch element. It is possible to improve the overall performance of the network by using a larger number of buffers ($b$) relative to the switch size ($d$). However,
an increase of \( B = b/d \) buffers \((B = \text{number of shared buffers per port of switch})\) will be at the cost of circuit complexity.

The availability of any buffer slot is determined by the switch element controller. If a particular requested output is free, the cell can be routed out so that it passes the switch through the output crossbar. Otherwise, it remains in the buffer until the control data of the cell in the controller part gets the necessary priority to win the contention. The cell circulation is controlled by the local buffer controller (LBC). There are two types of data transmitted from the controller to the data slices. Each buffer slot of a data slice receives a buffer-control bit telling it whether it should keep the cell for the next cycle or accept a new cell. The second type of information given to the data slices is the address (buffer row number) of the winning cell. These data are received by the data slices through a bus and are used to identify which buffer is allowed to release the current cell to the desired output.

3.3.3. Controller Unit

The controller part makes decisions leading to the transmission of a cell to the requested output(s). As shown in Figure 3.8, the controller receives only the headers (control information) of cells on its upstream data lines, ud(0) through ud\((d - 1)\). This occurs at the same time that the data slices receive the remaining data portion of the cells. In the controller, the header decoder (HD) first converts the control information of an arriving cell into a \( d \)-bit stream of initial request vector. This bit vector in fact carries the information pertaining to the replication of a cell so that any "1" bit represents a request for one of the \( d \) corresponding switch outputs. A complete discussion of the cell replication mechanism will appear in the next sections. The initial request vector gets routed through an input crossbar to the buffer control circuit (BCC). The BCC
generates a priority for each cell. This information along with the request vector enters an array of arbitration elements (AE). Each cell in a column of the AE array contends with other cells on a shared bus to access the switch output associated with that column.

After a cell wins the contention, its identity (buffer index number) is transmitted to the output control circuit (OCC). This identity and the buffer-control bit explained earlier are transferred to the data slices signaling them to release all m slices of the cell. This mechanism ensures that a losing cell remains in the buffer, and the BCC raises its priority by one so that it can participate in the next round of contention with a higher chance of winning. This process is repeated until eventually the cell wins. The identities of winning cells are transmitted to the data slices if downstream grant signals, dg(0) through dg(d – 1), from the downstream neighbors are active. The switch element controller in turn generates a corresponding set of upstream grant signals, ug(0) through ug(d – 1), which are sent to the upstream neighbors in the switching network.
An *upstream grant circuit* (UGC) generates the \(ug(i)\) signals whenever it is prepared to receive a cell.

### 3.3.4. Complexity of Switch Element

In this section, we estimate the complexity of a \(d \times d\) switch element each containing \(b\) shared buffer slots. The complexity of the data slice is:

\[
C_{ds}(d, m) \approx LdBc_1/m + 2Bd^2c_2, \tag{3.2}
\]

where \(B\) is the number of shared buffers per port of the switch element, \(L\) is the cell length in bits, \(c_1\) is the transistor count per bit of buffer memory, and \(c_2\) is the transistor count per crosspoint. For \(d = 16, B = 3, m = 9\) slices, and assuming \(c_1 = 12\), and \(c_2 = 24\) the total complexity of the data slice is estimated to be 67K transistors.

The complexity of the controller is estimated to be

\[
C_c(d) \approx d(c_3 + c_4 + c_5) + dBc_6 + B(c_2 + c_7)d^2, \tag{3.3}
\]

where \(c_3, c_4, c_5, c_6\) and \(c_7\) are the complexities of the HD, UGC, OCC, BCC, and AE, respectively. For \(d = 16, B = 3\), and assuming \(c_3 = 1K, c_4 = 2K, c_5 = 100, c_6 = 500,\) and \(c_7 = 100\), the total complexity of the controller is about 150K transistors.

### 3.4. Main Switch Functions

In this section, the main components of the switch element are discussed in detail. In particular, this section focuses on the techniques employed for buffering and multicasting cells, and explains the details of the contention resolution processes. At this point, we provide the actual switch specifications upon which the switching system was designed. These include the network size of \(n = 256\) with switch size \(d = 16\), the number of shared buffers within the switch \(b = 48\), and the number of data slices (data path) \(m = 9\). Some VLSI design details of the switch have also been presented in [35].
3.4.1. Buffer Control Mechanism

The buffer control circuit (BCC) is located at the heart of the switch controller as shown in Figure 3.8. The functional diagram of the BCC is depicted in Figure 3.9. It consists of four main parts: request vector circuit (RVC), waiting time generator (WTG), row number generator (RNG), and buffer-control bit circuit (BBC). For each cell, the RVC receives a serial 16-bit vector of cell multicast information on the \textit{ini.req.vec} line. The vector specifies the outputs to which copies of a given cell are to be sent. The RVC keeps this vector in an internal 16-bit shift register and generates an identical copy of this field on the \textit{req.vec} line as part of the data needed for the contention process. After the first cycle of the contention resolution, the RVC receives a vector containing the acknowledgments of allocated outputs from the arbitration elements on the \textit{all.ack} line. The RVC compares the \textit{all.ack} and \textit{req.vec} and determines which outputs need new rounds of contention. It then sends a copy of an updated \textit{req.vec} to the arbitration elements, and as long as the cell replication processes are not complete it, the cell must remain in the buffer. The \textit{req.vec} continues to be updated until all copies of the cell are successfully transmitted.

The RNG and WTG produce a 12-bit priority field on the \textit{req.priority} line. The combination of these two blocks is referred to as the request priority circuit (RPC). The WTG generates a 6-bit waiting time which is followed by another 6-bit field containing the row index of the buffer produced by the RNG. The waiting time is increased by one every time a cell loses the contention. When an arbitration element acknowledges and selects its corresponding bit among the 16 bits of the \textit{req.vec}, the WTG increases the cell waiting time if the cell loses the contention. This causes the BBC to generate a buffer-control bit. This bit passes serially through the next buffer control circuit and
Figure 3.9: Functional diagram of the buffer control circuit (BCC).

is finally sent out to the data slices as a control signal for buffers. The contention resolution process is presented in detail below.

3.4.2. Contention Resolution Process and Cell Replication

There is an array of 48×16 arbitration elements (AEs). As explained before, the buffer control circuit sends a 12-bit request priority (req.priority) and a 16-bit request vector (req.vec) to the corresponding row of AEs. The request vector specifies which outputs of the switch are requested. If only one of the 16 bits is "1," the case is a point-to-point connection, otherwise the vector configures multicast connections where each "1" corresponds to a request for a switch output. Figure 3.12 is a timing simulation for the contention resolution of two competing cells. This figure illustrates how cells are replicated within a switch element. The first cell indicated by req.vec0 is assumed to
Figure 3.10: A column of arbitration elements (AEs), and the contention bus structure.

be replicated to six different outputs: 0, 2, 5, 9, 14 and 15. The BCC to which a
cell belongs sends this 16-bit request vector on the req_vec line simultaneously to all
arbitration elements of the same row. The vector contains a logic “1” for each requested
output. The priority of a given cell is sent serially to the 16 arbitration elements of that
same row. Each AE passes on the request priority to the next AE; with one-bit delay.
This mechanism allows every AE to receive its requested bit in the req_vec on time when
the cell’s req_priority arrives and to select only the appropriate request bit among the
16 bits.

To understand the cell contention resolution in the simulation of Figure 3.12, con-
sider Figure 3.10. At a given column i of AEs, all req_vecs arrive at the same time. This
ensures that all arbitration elements of column $i$ select the requested bits for the corresponding output $i$ simultaneously. Each bit of the req_vec tells the AE whether or not it should contend. The arbitration element performs the contention resolution based on a state machine. There are two distinct states of idle and contending. The bus_pre signal precharges the bus every clock cycle. The contention at an arbitration element starts with the arrival of the first bit of the 12-bit field req_priority and continues until a cell completely accesses the shared bus vcb, or loses the contention. It should be mentioned that req_priority bits are sent by AEs serially to vcb, implementing a wire-NOR. During the contention, if one of the req_priority bits of a cell is "zero", while that of at least one of the competing cells is "one", the internal state machine causes the remaining bits of the req_priority to be blocked. Thus, the cell cannot continue the contention, and it loses the contention. But if the bit is not zero, the next bit is examined on the bus and so on. This continues until the cell loses or wins the contention.

Example. We now return to the timing simulation of Figure 3.12 as an example. Assume two cells being stored in buffer numbers 5 and 7 have already lost the contention 6 times and 4 times, respectively. Thus, the request priorities (waiting time + row number) of these two cells are req_priority0 = 000110-000101 and req_priority1 = 000100-000111, respectively. As a rule, the competition always starts at the leftmost of the req_prioritys, thus, it is obvious that the first cell wins the contention at the fifth bit. Note that the two internal signals, $a_{03.0}$ and $a_{03.1}$, which track the two req_prioritys (starting from the clock cycle 20) indicate, respectively, the winning and the losing conditions of the two cells. Consequently, after a 12-bit-long competition, only the cell with the highest priority wins the contention. During this process the content of the vcb bus is transmitted to the output control circuit (OCC). The OCC sends back an acknowledgment to the AE on the occ.ack by which the AE can provide the BCC a
corresponding winner allocation acknowledgment of all.ack. This data is moved on a shared acknowledgment bus.

3.4.3. Flow Control

The upstream grant circuit (UGC) generates a flow control signal for its upstream switch neighbor. Each controller receives a set of 16 downstream grants (flow control signals) from the downstream neighbors. The controller produces a corresponding upstream grant (ug_i) if it is prepared to receive a cell on the upstream data lines ud_i. The overview of the UGC is seen in Figure 3.11. Basically, the UGC uses two sets of data, 48 buffer-control bits and 16 upstream data. There is a single bit in the header of each cell called D/C which signals the presence of a cell. The arriving-cell counter (AC Counter) counts the number of arriving cells by adding up the D/C bits and asserts
the result on 5 lines of \( nac(n+1) \). The same operation takes place in the buffer-control bit counter (BB Counter) in order to add up the number of buffer-control bits and to generate a 6-bit wide \( neb(n+1) \). The number of required upstream grants is equal to \( neb(n+1) \) minus \( nac(n+1) \). A synchronous adder (SA) carries out this function. The appropriate number of grants, denoted by \( g \) in the figure, appears on a 6-line bus. The value \( g \) along with two internal values of index and the last generated point \( i \) are used in the grant signal selector (GSS) to select the eligible inputs to receive grant flow control signals.

3.5. Hardware Simulations Results

The layout design of the prototype switch element is based on 1.2\( \mu \) CMOS VLSI technology. Figure 3.12 shows a timing verification of the switch chips using VLSI circuit simulations. We used 2-phase clocking (indicated in the figure by \( \phi_1 \) and \( \phi_2 \)) running at the speed of 100 MHz, and with \( V_{dd} \) of 5V. We have explained this simulation in part in the previous sections and here we give an overview of the results. We simulated the chips for two different competing cells with different priorities whose control fields are shown as \( req\_priority0 \) and \( req\_priority1 \), respectively. As seen, since \( req\_priority0 \) has a higher waiting time of 000110 (starting from msb), compared to the other cell waiting time of 000100, it wins the contention, and its buffer identity (000101) which is the second part of the priority field appears on the output bus of the controller \( win\_id0 \) through \( win\_id5 \). Thus, only the winning acknowledgment associated with this cell (\( all\_ack0 \)) becomes active, while \( all\_ack1 \) remains inactive.

It should be mentioned that there have been a number of papers and documents such as [10] [23] [49] and [65] that provided useful directions and methods for high-speed circuits and practical switch designs.
Figure 3.12: Timing simulation of the test chip for two contending cells. The comparison of two internal signals $a_{03.0}$ and $a_{03.1}$ starting from clock cycle 20: the cell with the higher priority ($req\_priority0 = 000110-000101$) wins the contention.
3.6. Performance Evaluation

Since the early 1980s, several research groups have taken considerable interest in the performance study of switching networks under different queueing disciplines. One of the earliest widely-cited papers related to this topic was presented by Jenq [26] who described the queuing behavior of binary banyan networks with a single buffer at each switch input. In this paper the states of the two buffers in each switch element were assumed to be independent. Although this assumption can be a source of inaccuracies, the method provides a simple analysis for buffered packet switching networks. Later, Szymanski and Shaikh [53] extended Jenq’s method to switching systems constructed from switches with an arbitrary number of inputs and an arbitrary number of buffer slots.

Turner in [61] extended the previous work to cover the analysis of switching networks using switch elements with shared buffers. This required a new technique which was also applied to the switches with FIFO and bypass input buffering. Improvements to the shared buffer analysis were developed in [6] leading to a higher accuracy than the previous work.

In this section, the performance of the Beneš network for point-to-point connections using shared buffering is reviewed. This evaluation provides a baseline for comparison with analysis of alternative switch architectures that will be presented in later chapters. The following analysis is approximately the same as the analysis of delta networks presented in [61]. The main variables for the analysis are defined as follows:

\[ n: \text{ Number of inputs and outputs of the network; } \]
\[ d: \text{ Number of inputs and outputs of a switch element; } \]
\[ k: \text{ Number of stages, for the Beneš network } k = 2\lceil \log_d n \rceil - 1; \]
\( \alpha \): Number of buffer slots in an input buffer;

\( b \): Number of shared buffer slots in a switch element;

\( B \): Number of shared buffer slots per port of a switch element;

\( \rho \): Offered load to an input buffer;

\( \pi(s) \): Probability that an input buffer contains exactly \( s \) cells;

\( \pi_i(s) \): Probability that a shared buffer in stage \( i \) of the network contains exactly \( s \) cells;

\( \phi \): Probability that stage 1 switch grants permission to an input buffer for sending cells;

\( \phi_i \): Probability that a successor of a stage \( i \) switch grants permission to its predecessor for sending cells;

\( a_i \): Probability that a cell is available to enter stage \( i \) of the network;

\( p_i(j, s) \): Probability that exactly \( j \) cells enter stage \( i \) of the network when the switch is in state \( s \) at the beginning of the current cell cycle;

\( q_i(j, s) \): Probability that exactly \( j \) cells leave a buffer in stage \( i \) of the network when the switch is in state \( s \) at the beginning of the current cell cycle;

\( \lambda(\delta, s) \): Transition probability of a stage \( i \) switch from state \( \delta \) to state \( s \);

For any probability \( x \), we also let \( \bar{x} \) denote \( 1 - x \).

An input port buffer can be modeled using a Markov chain. Explicitly, an input buffer receives traffic at rate \( \rho \) and is granted permission from a stage 1 switch for releasing cells with probability \( \phi \). The probability that a cell is available to enter the switch is:

\[
\alpha_1 = 1 - \pi(0).
\] (3.4)
The steady state probability of different states of the Markov chain for the input buffer are computed iteratively using the following set of equations:

\[
\begin{align*}
\pi(0) &= \pi(0)\tilde{p} + \pi(1)\tilde{p}\phi, \\
\pi(1) &= \pi(0)\rho + \pi(1)(\tilde{p}\phi + \rho\phi) + \pi(2)\bar{p}\phi, \\
&\vdots \\
\pi(s) &= \pi(s-1)\rho\bar{\phi} + \pi(s)(\tilde{p}\phi + \rho\phi) + \pi(s+1)\bar{p}\phi, \\
&\vdots \\
\pi(\alpha-1) &= \pi(\alpha-2)\rho\bar{\phi} + \pi(\alpha-1)(\tilde{p}\phi + \rho\phi) + \pi(\alpha)\phi, \\
\pi(\alpha) &= \pi(\alpha-1)\rho\bar{\phi} + \pi(\alpha)\bar{\phi}.
\end{align*}
\]

Let \( Y_d(r, s) \) be the probability that a switch that contains \( s \) cells contains cells for exactly \( r \) distinct outputs. \( Y_d(r, s) \) is computed from the following recurrence:

\[
Y_d(r, s) = \begin{cases} 
1 & s = r = 0 \\
0 & (s > 0 \land r = 0) \lor s < r \\
\frac{d}{d}Y_d(r, s - 1) + \frac{d-1}{d}Y_d(r-1, s-1) & 0 < r \leq s \end{cases}
\]

Note this equation is an approximation based on the assumption that addresses of stored cells are independent. While it is true that cell addresses are independent when the cells arrive at a switch element, the contention resolution process introduces correlations which we neglected in this analysis. Next, we need an expression for flow control within the switch. Recall that \( a_i \) is the probability that any given predecessor of a stage-\( i \) switch has a cell for it. For \( i \geq 1 \):

\[
a_i \approx \sum_{0 \leq j \leq b} \pi_{i-1}(j) \left[ 1 - (1 - 1/d)^j \right].
\]
This equation is also based on the assumption that the addresses of cells in the buffer are independent. Let \( p_i(j,s) \) be the probability that exactly \( j \) cells enter a switch in stage \( i \) of the network when the switch is in state \( s \) at the beginning of the current cell cycle:

\[
p_i(j,s) = \binom{m}{j} a_i^j (1 - a_i)^{m-j}, \tag{3.12}
\]

where \( m = \min \{d, B - s\} \). As mentioned before, cells within the network can be transmitted to the next stage with grant flow control. Recall that \( \phi_i \) is the probability that a successor of a stage-\( i \) switch grants permission to its predecessor for sending cells. \( \phi_i \) is calculated by:

\[
\phi_i = \sum_{0 \leq h \leq d - 1} \pi_{i+1}(h) + \sum_{0 \leq h \leq d - 1} \pi_{i+1}(b - h) \frac{h}{d}. \tag{3.13}
\]

Using \( \phi_i \), we can express \( q_i(j,s) \), the probability that exactly \( j \) cells leave stage \( i \) of the network when the switch is in state \( s \) at the beginning of the current cell cycle:

\[
q_i(j,s) = \sum_{j \leq r \leq \min\{d,s\}} Y_d(r,s) \binom{r}{j} \phi_i^j (1 - \phi_i)^{r-j}. \tag{3.14}
\]

Let \( \lambda_i(\hat{s},s) \) be the probability that a switch in stage-\( i \) of the network contains \( s \) cells in the current cycle given that it contains \( \hat{s} \) cells during the previous cycle. This transition probability is computed by:

\[
\lambda_i(\hat{s},s) = \sum_{h = \max(0, s - \hat{s})}^{\min\{d, b - s\}} p_i(h, \hat{s}) q_i(h - (s - \hat{s}), \hat{s}), \tag{3.15}
\]

where \( 0 \leq (s \text{ and } \hat{s}) \leq b \), and
\[
\pi_i(s) = \sum_{\hat{s} = \max\{0, s-d\}}^{\min\{b, s+d\}} \pi_i(\hat{s}) \lambda_i(\hat{s}, s).
\]

(3.16)

By putting together all of the above parameters, the system throughput can be calculated. The throughput refers to the number of cells leaving the network per output link per cell cycle, and it is given by:

\[
T = \frac{1}{d} \sum_{j=0}^{d} \sum_{s=0}^{b} j q_{k}(j, s) \pi_{k}(s).
\]

(3.17)

Figures 3.13 and 3.14 give the results of the performance evaluation for the throughput \((T)\) when the traffic \((\rho)\) varies up to a maximum load of 1.0. In Figure 3.13, the network dimension is assumed to be \(n = 256\) with the switch size \(d = 4\). As one would expect, increasing the number of buffers per port \((B)\) increases the network throughput. It can be seen in this figure that all three curves flatten out after the offered load exceeds the system queueing capacity.

Figure 3.14 shows the throughput curves with the same conditions but a larger switch size of \(d = 16\). The effect of using larger switches is apparent on all three curves. The improvement of the network performance with 16-port switches is mainly due to the reduction in the number of required stages when using larger switches.

In the next set of plots given in Figures 3.15 and 3.16, the effect of the network size on the network throughput is evaluated. In Figure 3.15, the network dimension is varied up to 256. The offered load is assumed to be 90\%, and the number of buffers per port of each switch element is fixed at \(B = 2\). The plots for switch element sizes, \(d = 4\) and \(d = 16\), demonstrate consistency of throughput when the network size increases.
Figure 3.13: Throughput ($T$) vs offered load ($\rho$) when the switch size $d = 4$ and number of shared buffers per port of switch ($B$) varies.

Figure 3.14: Throughput ($T$) vs offered load ($\rho$) when the switch size $d = 16$ and number of shared buffers per port of switch ($B$) varies.
Figure 3.15: The effect of enlarging network size (n) on throughput (T) for two different switch sizes (d) where the number of buffers per port of switch $B = 2$.

Figure 3.16: The effect of enlarging network size (n) on throughput (T) for two different switch sizes (d) where the number of buffers per port of switch $B = 3$. 
Figure 3.17: Complexity of the network per port (in million transistors) vs the network size \( n \) and the number of buffers per port \( B \).

Figure 3.16 gives similar results but with \( B = 3 \). As expected, a larger number of buffers increases the system throughput.

3.7. Evaluation of Complexity

The total complexity within the switch fabric, \( C_{\text{net}} \), is a combination of complexities within the copy network, BTCs, and routing network which are denoted by \( C_{cn} \), \( C_{btc} \), and \( C_{rn} \), respectively; thus \( C_{\text{net}} \) is:

\[
C_{\text{net}}(m, n)/n = (C_{cn} + nC_{btc} + C_{rn}) + n(C_{ipp} + C_{opp}).
\]

Assuming every switch element has \( m \) data slices according to Figure 3.7 and using the total number of switch elements given in Equation 3.1, clearly \( C_{cn} = C_{rn} = n/d(2[\log_d n] - 1)(mC_{ds} + C_c) \); thus, the complexity per port of the switch is:

\[
C_{\text{net}}(m, n)/n = \frac{2}{d} \left[ \frac{n}{d} (2[\log_d n] - 1)(mC_{ds} + C_c) \right] + C_{btc} + (C_{ipp} + C_{opp})
\]

\[
= \frac{2}{d} \left[ (2[\log_d n] - 1)(mC_{ds} + C_c) \right] + C_{btc} + (C_{ipp} + C_{opp}).
\]

(3.19)
For a switching network of $n = 256$, $d = 16$, $m = 9$, $B = 3$, $C_{\text{net}}/n \approx 3.52M$ transistors.

Note that the values for the complexities of the data slice and the switch controller can be obtained from Equations (3.2) and (3.3). Also, in the above equation we assumed $C_{\text{bt}} = 2K$ transistors and $C_{\text{iop}} + C_{\text{opp}} \approx 3.2M$ transistors. Figure 3.17 shows the switch fabric complexity per port for different network dimensions ($n$) and number of buffers per port ($B$).

It should be noted that the transistor count is used in this section as a rough indicator of system cost. A more careful analysis would also take into account the layout area for the circuitry and other engineering factors, e.g., the design and size of the required printed circuit board.

3.8. Remarks

The switching architecture discussed in this chapter is designed with the intention of implementation using VLSI technology. In the next chapter, we will evaluate this architecture from a different technological perspective.
4. **EVALUATION OF WSI FOR SWITCHING SYSTEMS**

While today's integrated circuit technologies can meet the demands of the computer industry, *wafer-scale integration* (WSI) in *monolithic* and *hybrid*\(^1\) forms has been advancing at an astonishing pace toward higher circuit densities and higher speeds. Wafer-scale technology can offer significant potential advantages such as high reliability and performance with reduced interconnection propagation delay and power dissipation.

Studies concerning the design and implementation of switching systems, so far, use VLSI technologies in which components are fabricated on a wafer, and then separately packaged. In this chapter, we consider the possibility of using wafer-scale integration technology for ATM switching systems. This technology is well-suited to the development of switching systems due to the minimization of interconnection lengths in a wafer-scale system.

This chapter is organized as follows. In Section 4.1, we review the concept of wafer-scale technology and summarize integrated circuit yield models. In Section 4.2, the cost of interconnections in a typical multistage network is evaluated and the possibility of using WSI technology for ATM systems is discussed. We base the analysis on the switch architecture described in the previous chapter and present a modified version of the architecture as a primary option in Section 4.3 and assess its complexity in Section 4.4.

\(^{1}\)hybrid WSI is usually called *multichip module* (MCM)
4.1. Wafer-Scale Integration

The earliest reference to wafer-scale integration is in 1964 on 1 inch diameter wafers according to [30]. Little significant work was carried out until the late 70's. Since then many research groups have expressed interest in this concept. In 1987, fabrication of nearly 2 million transistors on a single chip was announced. Now, we can expect the imminent application of wafer-scale integration technology to provide a gigantic integrated circuit on a single wafer with full connectivity among all neighboring modules\(^2\). The average circuit complexity of a single wafer may soon reach more than 1 billion transistors. Since the integrated circuit modules do not always function due to defects, the system design must allow all the modules to be tested independently and the properly functioning modules to be configured to form a working system. The earliest motivation for WSI was the need for more computing power. Also, producing higher circuit speed is generally easier using WSI because of the reduction in off-module connections. [7] is a good example of a recent experimental wafer-scale system. The potential advantages of WSI are as follows:

- Increased speed, since off-module connections are minimized.
- Reduced transmission line reflection of signals, due to shorter interconnections.
- Reduced power consumption, due to removal of most modules output pad drivers.
- Reduced area, since individual packaging for modules within the wafer is no longer needed.

\(^2\)We interchangeably use module or unit to refer to the basic building block used to construct a wafer-scale system. These units are comparable in complexity to modern VLSI components and are replicated many times on a wafer.
• Reduced weight and volume of the system.

• Reduced overall production cost, since individual packaging and assembly costs are removed.

• Increased reliability since there is no need for connectors and bonding for chip-to-package, package-to-pcb, and inter-pcb connectors.

• Fault tolerance features of WSI devices may allow for system reconfiguration in the field.

However, there are a few drawbacks with WSI. The main one is the need to design the system so that its building blocks can be tested and then the defect-free modules configured to produce a working system. The inclusion of additional circuitry for defect tolerance partially offsets the area reductions cited above. Also, heat removal in a wafer-scale system may require special cooling equipment. Another drawback can be a restricted range of application since complex WSI systems may lack the broad applicability of conventional integrated circuits.

4.1.1. Defect Tolerance and System Reconfiguration

Due to the existence of defective modules, not all fabricated modules on a wafer can be used. Figure 4.1 shows an illustrative example of how failing modules get bypassed. Efficient algorithms for bypassing the defective modules must be developed to achieve a high yield. This also adds a harvesting factor [7] to the list of considerations. Between each pair of adjacent modules there are a number of bypassing switches as shown in Figure 4.1. Separate test circuitry controls the operation of these switches. Also, a number of spare modules are required to replace nonfunctioning modules. The number of spare modules required depends on the yield. After all modules are tested, the
Figure 4.1: Method of bypassing defective integrated circuit modules in WSI.

entire wafer has to be reconfigured in order to produce a working system. In the illustrative example of Figure 4.1, integrated circuit units are designed to communicate vertically (north, south) and horizontally (east, west). Note that, in this example, a target harvesting of 2 modules per row (i.e., from node $R_i$ to $R_{i+2}$) and 2 modules per column (i.e., from node $C_i$ to $C_{i+2}$) was assumed. Also notice that in the second row a functioning module which belongs to the spare column is bypassed due to the mentioned target size. In order to reconfigure a wafer-scale system, the concept of yield and distribution of defects must be studied.

4.1.2. Yield Analysis

In this section, we briefly compare the two most commonly used yield models, the Poisson distribution and negative binomial distribution. The Poisson model assumes that the spatial distribution of defects is random [51]. In this model the probability of
$j$ defects occurring within one module is given by:

$$Pr \{X = j\} = \frac{e^{-\nu} \nu^j}{j!},$$

(4.1)

where $\nu$ is the mean number of defects per module. $A$ and $D$ are the module area and defect density respectively, and $\nu$ is given by $\nu = A \cdot D$. Yield ($y$) is defined to be the probability of no defects on the module ($j = 0$):

$$y = Pr \{0 \text{ defect}\} = e^{-A \cdot D}.$$  

(4.2)

This model is based on two underlying assumptions: the defect density is constant, and the occurrence of a defect at any location is independent of any other defects and is thus random. These assumptions are reasonable if the area of each individual module is not too large. One practical example of this model was presented in [63] where the Poisson model is used to evaluate the yield of a wafer-scale memory array. For small-area modules, the Poisson distribution model was found to be acceptable since inter-module variations of defect density were not apparent. The variations become obvious when larger modules are fabricated. The assumption of random distribution of defects for larger modules gives a pessimistic yield value. A significant modification of the Poisson model was developed by Murphy [45, 51]. He observed that the defect density $D$ is not constant and may vary from module to module on a wafer, and even from wafer to wafer, or wafer lot to wafer lot. This observation resulted in $D$ being expressed as a random variable with probability density function $f(D)$. Murphy suggested the triangular Gaussian distribution, leading to the following expression for yield:

$$y = Pr \{0 \text{ defect}\} = \int_0^\infty e^{-A \cdot D} f(D) dD \cdot$$

(4.3)

Finding an appropriate $f(D)$ (compounder for the Poisson distribution) was the next challenge. Several different compounders have been suggested to achieve an accurate
model, among them in [50], use of the *gamma distribution* for \( f(D) \) in Equation (4.3) has been proposed which results in the *negative binomial distribution*:

\[
y = \frac{1}{1 + \frac{A_D D_0}{(\sigma/\mu)^2}},
\]

(4.4)

where \( \sigma^2 \) and \( \mu \) are the variance and *mean* associated with the gamma distribution, respectively, and \( D_0 \) is the mean number of defects per unit area. Expression (4.3) has been successfully used to model integrated circuit yields. We will use this model in the next chapters to evaluate the yield of the proposed wafer-scale switching systems.

4.2. Cost of Interconnections in Multistage Switching Networks

In this section, we evaluate a typical multistage switching network for WSI implementation. Consider the design of an \( n \times n \) Beneš network using \( d \times d \) switch elements that was presented in Chapter 3. According to Equation (3.1), the Beneš copy network is constructed with \( 2[\log_d n] - 1 \) stages.

Consider the routing complexity created by the Beneš copy network \( (B_{n,d}) \) presented in Figure 3.4. One way to efficiently lay out interconnections of a \( B_{16,4} \) network is illustrated in Figure 4.2. In this figure we show wiring tracks of the first stage of \( B_{16,4} \). Although there are \( n = 16 \) vertical wiring tracks, this number can be further reduced to 8 by laying out two wires per track. Let \( f(n,d) \) represent the total number of vertical wiring tracks in a Beneš network \( B_{n,d} \). Then:

\[
f(n,d) = 2 \left[ (m+1) \left( \frac{n}{2} + \frac{n}{2d} + \frac{n}{2d^2} + \cdots + \frac{d^2}{2} \right) \right],
\]

(4.5)

where each track consists of \( m \) data paths and a flow control line. By substituting
Figure 4.2: Realization of the wiring in the first stage of a Beneš network.

\[ n = d^k \] in the above equation, we have:

\[
f(n, d) = (m + 1) \left( d^k + d^{k-1} + d^{k-2} + \cdots + d^2 \right)
\]

\[
= (m + 1) \left( \sum_{i=0}^{k} d^i \right) - (d + 1)
\]

\[
= (m + 1) \left( \frac{d^{k+1} - 1}{d - 1} - (d + 1) \right)
\]

\[
= (m + 1) \left( \frac{d^{k+1} - d^2}{d - 1} \right). \quad (4.6)
\]

Assume one metal layer each with \(8\lambda\) pitch to be used for vertical wiring (where \(\lambda\) is 1/2 the minimum feature size), then the total width of the wiring channel is:

\[
W = \left[ (m + 1) \left( \frac{d^{k+1} - d^2}{d - 1} \right) \right] 8\lambda + W_s, \quad (4.7)
\]

where \(W_s\) is the total width of spare wiring tracks required for reconfiguration. With \(d = 2, k = 8, m = 9\) and \(\lambda = 0.5\mu m\), \(W\) is \(20320\mu m + W_s\) which is equal to \(2.03\)cm + \(W_s\). Note that these figures are just for the copy network, without taking into account two other main parts the BTCs and the routing network. By including interconnections of the routing network the total width is increased to \(4.06\)cm + \(2W_s\). Larger switch
elements in a network could decrease $W$ since the total number of stages is reduced, but this can only be done at the risk of a lower switch element yield. Clearly, yield is a non-linear function of area according to Equation (4.4). For example, the switch element presented in the previous chapter using $d = 16$ requires 760K transistors. We remember that this complexity is still based on the bit-slice architecture which is motivated by reducing the switch complexity for VLSI implementation. We also remember that, as $n$ gets larger, area grows non-linearly as seen in Equation (4.7) and thus to provide a defect-free system, a larger number of spare interconnections are needed.

Although the cost of testing interconnections can be high because of irregularity in the interconnection layout, a multistage switching network can still be an option for the implementation using WSI technology. In the next sections and chapters, we propose alternative designs taking advantage of WSI capabilities for reducing the cost of interconnections and testing.

4.3. A Modified Version of Multi-Stage Network for WSI

The elimination of internal bondings, pads, and long inter-module connections in a wafer-scale system gives us the ability to modify the multistage switches for WSI in ways that would not be practical in a system using VLSI components. The previous section indicated that a large area on a wafer must be committed just for interconnections using a conventional multistage architecture. This drawback motivates us to modify the structure of the gigabit ATM system presented earlier, to allow the use of WSI with reduced interconnection area.
Figure 4.3: A modified network architecture using shared buffer crossbars.

4.3.1. Network Controller and Delay Issue

The shared buffering structure of the switch element described in Chapter 3 is utilized to form the entire copy or routing network on a larger scale (see Figure 4.3). This leads to replacement of multistage Beneš networks with uniform shared buffer crossbar networks (SBC). Each one of these two SBCs uses the bit-slice structure. In each SBC network, thus, there are $m$ identical network data slices (NDS) and a single network controller. The structure of each NDS is similar to the data slice of the switch element presented earlier, but with $n$ inputs instead of $d$, and $b_s$ shared buffers instead of $b$ ($b = d \cdot B$, $b_s = n \cdot B$), where $b$, $B$, $d$ and $n$ are the total number of buffers, number of buffers per port, the switch dimension, and the network dimension, respectively, in the previous architecture.

The network controller is a uniform integration of all the individual controllers originally designed for the switch elements. The network controller receives the cell headers from all $n$ inputs. Since $n$ inputs are supposed to contend simultaneously for access to the desired outputs, it would take at least $n$ clock cycles for the controller to arbitrate which cell should win the competition if it were implemented as described before. Assume $n = 256$ and every cell is 55 bytes long; a contention resolution process for this example would require $256/55 \approx 4.6$ cell cycles. This long processing
Figure 4.4: The copy network using a bit-slice *shared buffer crossbar* structure.
Figure 4.5: The uniform structure of the network data slice designed for WSI.

delay constraint can be overcome by modifying some the controller. Let \( n \) columns of arbitration elements be divided into \( p \) groups each with \( d \) columns, say columns \((x_0, \ldots, x_{d-1}), (x_d, \ldots, x_{2d-1}), \ldots, (x_{(p-1)d}, \ldots, x_{n-1})\). As long as the ratio of \( d = n/p \) satisfies:

\[
\frac{n}{p} < \text{(a cell cycle length),}
\]

there can be two options to reduce the contention resolution delay. The first option: while the network controller is kept in one slice, to send the request vectors from the BCCs to \( p \) individual lines each of which is received by one of the \( p \) corresponding groups of arbitration elements. In this manner, the contention resolution duration would be reduced by a factor of \( p = n/d \). The drawback to this scheme is the high complexity of the lines exiting the BCCs, but it would guarantee the objective. An alternative way to reduce the cell contention time is to replace the request vectors with the starting and ending column numbers of the arbitration elements. Each one of \( p \) groups of arbitration elements examines these numbers and determines whether each number belongs within
Figure 4.6: Complexity of the network per port (in million transistors) vs the network size \((n)\) and the number of buffers per port \((B)\).

It. Thus, only corresponding starting and ending numbers are passed and processed through the group of arbitration elements. This second method, however, requires a more complex arbitration element.

4.4. Evaluation of Complexity

The transistor count for each network data slice is denoted by \(C_{nds}\) and is given by:

\[
C_{nds}(m, n) \approx L n B c_1/m + 2 B n^2 c_2, \tag{4.9}
\]

where \(c_1\) and \(c_2\) are the complexities per bit of memory and crosspoint, respectively. \(C_{nc}\) is the cost for the network controller and is given by:

\[
C_{nc}(n) \approx n(c_3 + c_5) + n B c_6 + B(c_2 + c_7)n^2, \tag{4.10}
\]

where \(c_3, c_5, c_6, c_2, \) and \(c_7\) are the transistor counts for the HD, OCC, BCC, crosspoint, and AE respectively. The overall cost for the switch fabric, \(C_{net}(m, n)\), is:

\[
C_{net}(m, n) = (C_n + n \cdot C_{blc} + C_{rn}) + n(C_{ipp} + C_{opp}), \tag{4.11}
\]
where $C_{ipp}$ and $C_{opp}$ are the complexities of the IPP and OPP, respectively. The copy and routing networks are identical, thus $C_{cn} = C_{rn} = [mC_{nds}(m, n) + C_{nc}(n)]$. Considering $m$ network data slices for each SBC network, the complexity of the switch fabric per port is estimated to be:

$$C_{net}(m, n)/n = \frac{2}{n} [mC_{nds}(m, n) + C_{nc}(n)] + C_{btc} + (C_{ipp} + C_{opp}).$$  \hspace{1cm} (4.12)

For $m = 9$, $k = 16$, $n = 256$, $B = 3$, and assuming $c_1 = 12$, $c_2 = 24$, $c_3 = 1K$, $c_4 = 2K$, $c_5 = 100$, $c_6 = 500$, $c_7 = 100$, $C_{btc} = 2K$, and $C_{ipp} + C_{opp} \approx 3.2M$, $C_{net}(m, n)/n \approx 4.86M$ transistors. Figure 4.6 shows the switch fabric complexity per port for different network dimensions ($n$) and the number of buffers per port ($B$).

4.5. Remarks

In this chapter we have modified the structure of the gigabit switch to obtain a reasonable architecture for WSI. The use of the uniform structure in the modified version rather than the interconnection network would potentially allow this architecture to carry more traffic than a multistage network. The main reason for this improvement is attributed to the fact that the switch has only one stage of routing where a cell always encounters queueing once. However, this structure still lacks the regularity which is a requirement for a simple defect tolerance strategy. The diversity of modules within a wafer would lead to a complicated system reconfiguration scheme. Specifically, This architecture would need as many testing steps as there are different modules on the wafer. In the next chapter, a simpler defect-tolerant and more practical wafer-scale switching network consisting of identical modules is proposed.
5. A MULTIPATH BUFFERED CROSSBAR FOR WSI

This chapter\(^1\) presents the design and analysis of an expandable wafer-scale switching architecture constructed with identical building block modules. This type of structure makes it easier to achieve simple system reconfiguration. We consider this architecture for high-speed ATM applications supporting multipoint communications. The switch is composed of a large multipath crossbar which can be easily scaled up or down according to the available choice of wafer size. The crossbar consists of buffered crosspoints each with two switches. With this configuration, incoming cells need to pass through only two stages of queueing to reach their desired output ports. However, it is desirable to keep the overall complexity of the architecture low in order to fit the largest possible network on a single wafer.

This chapter is organized as follows. In Section 5.1 the system architecture is introduced. Section 5.2 presents an analytical queueing model and uses it to evaluate the performance of the network. In Sections 5.4, 5.5, and 5.6, complexity, yield, and area of the switching system are assessed.

5.1. System Architecture

Figure 5.1 shows the overview of a switching system fabricated entirely on a single wafer. A multipath buffered crossbar network (MBC) as a switch fabric is introduced. The MBC has a scalable structure consisting of identical building block units. The strategy of using identical units simplifies the yield analysis and the system reconfiguration. The regularity in the structure of the network is made possible by the removal of the global

\(^1\)This chapter has been published in part in [36], [38], [43].
crossbar controller. A simplified controller is therefore included at each crosspoint. Input and output port buffers (including their processors) can be fabricated on the space available at two sides of the wafer as shown. Both input and output port buffers need spare units as shown in the figure to replace defective ones. For multicasting cells, the cell recycling method introduced by Turner [62] is used. This method eliminates the need for separate copy and routing networks and the need for broadcast translation circuits. It does require that the network have the ability to copy a cell to two distinct outputs and that each output port processors be linked to its input port processor by a recycling data path.

5.1.1. Design of a Multipath Buffered Crossbar

Conventional $n \times n$ crossbars are vulnerable to faults. A Multipath Buffered Crossbar (MBC) is proposed to reduce the probability of disconnection due to defects between each pair of input/output ports. The MBC network is a crossbar with $n$ rows and $k$
columns. Each row contains an input bus and an output bus. A cell being transferred from an input $i$ to an output $j$ is sent by input port processor $i$ on input bus $i$ to one of the $k$ columns. It then passes along this column to the output bus in row $j$. The crosspoints that make up the MBC network include mechanisms that allow the inputs to contend for access to the different columns. The crosspoint buffers in one row act as a distributed output buffer for the corresponding output port. As shown in Figure 5.2, in every crosspoint there are two different switches and a buffer.

Unlike a conventional crossbar, there are $k$ different paths in the MBC network to establish a connection between a particular pair of input and output ports. The network also has the capability to connect any input $i$ to all or any subset of outputs. For now, consider a case of point-to-point connection between input $i_1$ to output $a_1$. The cell is first randomly sent to a crosspoint at one of the $k$ shared data buses, say column $(j - 2)$. In case the crosspoint on this bus is not available for any reason, another
crosspoint selected at random is examined and so on. The reason for randomizing the initial location is to distribute the traffic as evenly as possible. When a functioning crosspoint (crosspoint 3) at column \( j \) is selected, the cell can be sent to a second crosspoint (crosspoint 4) at this column, unless the selected crosspoint is found to be faulty or its buffer is full. The cell is buffered in crosspoint 4 and can be sent out after a contention resolution process in its row.

In the case of multicast connections, as will be described later, we use the cell recycling technique [62] in which a multicast connection is converted to several bicast connections implemented one at a time within the network. Referring to Figure 5.2, assume a cell at input \( i_1 \) is to be copied to a set of two outputs \( p = \{ a_1, a_2 \} \). A bicast connection is treated as two independent point-to-point connections. In other words, sequential transmission of two cells is carried out recursively and independently. Since a selected crosspoint is not always available, any unfinished connection may be completed on another randomly selected column such as \( (j + 1) \). However, the following three methods are suggested for routing:

1. **Sequential transmission.** Copy the cell at the input buffers sequentially, treat the two copies independently, and complete recursively any unfinished connection on another randomly selected column like \( (j + 1) \) as shown in Figure 5.2.

2. **Copy twice or not at all.** Send the cell to two selected crosspoints of a column simultaneously so that it requires both to succeed if two acknowledgments are received, or neither if fewer than two acknowledgments are received. In the case of failure, the cell is retransmitted on another column.

3. **Copy as many as possible.** Send the cell to two selected crosspoints of the same column simultaneously, so that either the connections are successful, or unfinished connections are attempted on another column.
The implementation of the second method is potentially the simplest. Method 1 can offer a reasonably low complexity due to the simplicity of receiving acknowledgments in the sequential way. The third alternative method has the potential to perform better, but it requires more hardware compared to the other two. Thus, we base the analysis in this chapter on the sequential transmission of Method 1.

The availability of each crosspoint at a given node \((i, j)\) is dynamically reported to the input ports by a flag \(g_{i,j}\). If more than one input port requests a given column, based on a contention resolution process, only one of them can use the column. Cells in the other input ports receive higher priorities to examine other columns. Obviously, the more the cell receives priority increments, the sooner it is granted one free column that it selects randomly. Once the cell gets accepted by one of the \(\beta\) buffers of a second crosspoint, it contends to get access to its corresponding output. The losers of the contention resolution get higher priorities for the next contention cycle until they are successfully transmitted. In Figure 5.2, the connections \(i_1 \to a_1\) and \(i_1 \to a_2\) are made possible through nodes \(\{2, 4\}\) and nodes \(\{3, 6\}\) respectively.

### 5.1.2. Input and Output Port Processors

Figure 5.3 shows the basic operation of the input port processors (IPP) and the output port processors (OPP). In [62], Turner introduced an optimal nonblocking multicast virtual circuit switching system in which the multicast implementation is carried out through a novel method called cell recycling. At each IPP, a virtual circuit translation table (VXT) provides two pairs of 'output address, and virtual circuit identifier (VCI)' plus two additional bits indicating whether or not the cell should be recirculated. Based on the multicast tree mechanism explained in [62], the cell copy at either or both of the
outputs can be recycled to its corresponding input processor in which another pair of copies is similarly made. This sequence continues until all copies are sent out.

Figure 5.3 illustrates the operation in which a sender can send three copies of a cell to three different receivers. A cell entering at input $i_1$ with VCI $i$ is duplicated and forwarded to output $a_1$ with VCI $d_1$ and to an arbitrary output $a_r$ with VCI $d_r$. The cell at $a_r$ is recycled, and its VCI 'd_r' is used to select a new table entry from VXT2. Based on the information available in VXT2, the cell at the corresponding input port $i_2$ is forwarded to output $a_2$, VCI 'd_2' and output $a_3$, VCI 'd_3'. At every IPP, a receive buffer (RCB) holds cells from the input link, and a recycling buffer (RCYC) keeps cells from the recycled line. Both buffers serve as the waiting queues for cells entering the switching network. Because the network may deliver cells in a different order, a resequencing buffer (RSQ) at the output port processors restores the proper ordering. The cells waiting to be transmitted to the outgoing link are held in the transmit buffer (XMB) after they have been resequenced.
5.1.3. Structure of a Self-Driven Crosspoint

As an option for each crosspoint of the MBC network, a self-driven crosspoint (SDC) shown in Figure 5.4 is proposed. In the MBC network, a connection between an input and output port is made through two data buses, three address buses, and two flag lines. To introduce the other parts of the module, we describe the column contention process and the row contention process for setting up a connection through two crosspoints.

**Column Contention Process.** The input port buffer sends the address of the crosspoint on the column address bus. A column decoder (CD) scans this bus and compares the address of the requested crosspoint with the crosspoint index value. The equality of these two values implies a request for the corresponding crosspoint. If the crosspoint is found to be functioning, a row flag line activates and switch.a is closed to connect the input port processor to the column. A contention resolution process takes place through controllers of all input port buffers that select the same column. Column contention buses \( .a \) and \( .b \) carry the required information to resolve the contention. When this primary process is completed, the input port processor sends the output address on the output address bus.a and then through switch.a on the output address bus.b. This address is decoded by the output port decoder (OD). Similarly, the availability of the second requested crosspoint is acknowledged back to the input side of the fabric through a column flag line which is connected to the flag line of the first crosspoint. Once the input port processor receives the flag, it releases the ATM cell. The cell is buffered in the second crosspoint. The cell route consists of the input data bus, switch.a of the first crosspoint, and the column data bus. For a second crosspoint whose acknowledgment is not received, the same procedures are repeated on the next randomly selected column. The trials continue until all of the connections are completed. The column contention circuitry at the input port side of the network
Figure 5.4: The structure of a self-driven crosspoint for the proposed MBC network.

(not shown in the figure) is identical to the output contention circuitry which will be explained in the following section.

Output Contention Process. At each crosspoint there are $\beta$ buffers, each with $m$ slots, along with their local buffer controllers (LBC) to receive and hold the ATM cells. A priority field comprising the cell waiting time followed by an index number is produced by a priority field circuit (PFC) in order to specify the priority value for the cell. This field, starting with the msb of the waiting time, is sent to the output contention bus through the contention resolution circuit (CRC). The cell then enters a competition process to access the desired output. The target for all the cells in this competition is the output data bus. If the value indicated from the bus is identical to the one at the output of the PFC, the cell has won the contention. The corresponding comparator ($\text{comp}$) at its output ($a = b$) allows $\text{switch}_b$ to proceed to the switching operation; otherwise, at output ($a \neq b$) the comparator makes the Waiting Time Generator (WTG) increase.
the cell's priority, and the cell is kept in the buffer slot for the next round of contention but with a higher priority this time.

5.1.4. Defect Tolerance and WSI Reconfiguration

The special structure of the network allows the induction of a simple bypassing method for the defective chips. At the fabrication time, crosspoints are not connected to busses. Crosspoints are tested individually using RVLSI ( Restructurable VLSI) technology with the laser-programmed antifuse technique described in [2]. If a crosspoint is found to be defective, it is isolated from the network like crosspoint 1 shown in Figure 5.4. All functioning crosspoints are connected to links and buses using laser equipment.

With this system configuration, defective modules do not participate in the routing and the contention process but all links or control busses passing by with faulty crosspoints can be used to route cells to other crosspoints. This is a consequence of the fact that crosspoints must acknowledge each cell, using a flag line. This can be implemented by precharging the flag line and having crosspoints pull the flag line low to acknowledge a cell. If a defective crosspoint is requested by a cell, no acknowledgement is given as in the case in which a crosspoint is being used for another cell. Consequently, a faulty crosspoint simply reduces the effective traffic capacity of the network slightly, but has no effect on other crosspoints in its row or column.

5.2. Performance Evaluation

In this section, the performance of the MBC network is evaluated. To achieve better accuracy, the states of buffers in each queue are assumed not to be independent [61] as was explained in earlier chapters.
Figure 5.5: A queueing model for the multipath buffered crossbar.

5.2.1. Queueing Model

The queueing model for this system is shown in Figure 5.5. Cells enter the network from \( n \) different input port buffers and pass through any of the \( k \) crosspoints belonging to the same row as the input. As discussed earlier, these crosspoints do not contribute in any queueing operation and act only as ordinary connectors. Each cell exiting these nodes can confront \( n \) crosspoint buffers of length \( \beta \) that are situated on one of the \( k \) specified columns. A cell would select a buffer based on the desired address. In this system architecture, each crossbar row contains \( k \) buffers constructing a related group of buffers that we refer to as a queue-set as seen in the figure. The winning cell of each set is then permitted to enter the corresponding output port buffer. The main variables for the analysis are defined as follows:

\[
\begin{align*}
\text{n:} & \quad \text{Number of input/output ports} = \text{number of queue-sets;}
\text{k:} & \quad \text{Number of columns} = \text{number of buffers per queue-set;}
\text{\( \alpha \):} & \quad \text{Number of buffer slots in an input port buffer;}
\text{\( \beta \):} & \quad \text{Number of buffer slots in a crosspoint buffer;}
\end{align*}
\]
\( \gamma: \) Number of buffer slots in an output port buffer;

\( b: \) Total number of buffer slots in a queue-set, \( b = k\beta; \)

\( \rho: \) Offered load to the input port buffers;

\( \pi_i(s): \) Probability that an input port buffer contains exactly \( s \) cells;

\( \pi_x(s): \) Probability that a queue-set contains exactly \( s \) cells;

\( \Psi: \) (Probability that a cell forwarded by input port buffer \( i \) is admitted by a "given" crosspoint on column \( j \) = (Probability that input buffer \( i \) receives the final flag acknowledgment indicating that the requested buffer on column \( j \) is available);

\( a: \) Probability that a cell is available to enter the network;

\( a_x: \) Probability that a cell is available to enter a queue-set;

\( p(j, s): \) Probability that exactly \( j \) cells enter a queue-set when the queue is in state \( s \) at the beginning of the current cell cycle, \( j \in \{0, 1, \ldots, k - 1\}; \)

\( q(j, s): \) Probability that exactly \( j \) cells leave a queue-set when the queue is in state \( s \) at the beginning of the current cell cycle, \( j \in \{0, 1\}; \)

\( \lambda(\hat{s}, s): \) Transition probability of a queue-set from state \( \hat{s} \) to state \( s \);

\( f: \) fraction of the cells with fanout 2 (bipoint connections).

The queueing behavior of this system is analyzed considering the fact that the state of each buffer within a queue-set is dependent on the state of the other buffers in the queue-set. There will be no grant flows between the output port buffers and the crosspoint buffers, while the admission of a cell by a queue-set is made possible by granting the flag acknowledgments to the input port buffers. It should also be noted that the analysis that follows assumes that all crosspoints are functional.
Let \( z_k^\beta(s) \) be the number of ways to distribute \( s \) cells among \( k \) distinct buffers within a queue-set, under the restriction that each buffer may contain at most \( \beta \) cells. Note that here we are not concerned about the number of combinations of different cells in the queue. Assuming this, \( z_k^\beta(s) \) can be recursively computed by:

\[
z_k^\beta(s) = \begin{cases} 
1 & \text{if } (k = 1 \land s \leq \beta) \lor (s = 0) \\
0 & \text{if } s > k\beta \\
\sum_{0 \leq i \leq \min(\beta,s)} z_{k-1}^\beta(s-i) & \text{if } 0 < s \leq k\beta.
\end{cases}
\]

(5.1)

Let \( X_k^\beta(r,s) \) be the probability that a given crosspoint buffer has \( r \) cells when the entire queue-set contains \( s \) cells. Then

\[
X_k^\beta(r,s) = \frac{z_{k-1}^\beta(s-r)}{z_k^\beta(s)}.
\]

(5.2)

To further extend this analysis, an expression for the flow control between a crosspoint queue-set and the input buffers is required. A second crosspoint generates a final flag acknowledgment \( \Psi \) after having completed a three-step process. Figure 5.6 summarizes the signaling of these three steps for the transmission of each single cell. First, a crosspoint is requested, then, the corresponding column needs to be granted to the cell based on the contention resolution, and finally the second crosspoint to which the cell is addressed must be functional and must contain an empty buffer slot. These three statements can be combined to form \( \Psi \). The probability that a cell is available to enter the network, \( a \), is the same probability that the input port buffer is not empty and is given by:

\[
a = 1 - \pi_i(0).
\]

(5.3)

Let \( \psi \) be the probability that a cell contending for access to a column wins the contention. Note that the probability that any given input attempts to contend for any
given column is \(a/k\). Hence,

\[
\psi = \sum_{0 \leq c \leq n-1} \frac{1}{c+1} \binom{n-1}{c} \left(\frac{a}{k}\right)^c \left(1 - \frac{a}{k}\right)^{(n-1)-c},
\]  

(5.4)

where \(\binom{n-1}{c}\) is a binomial coefficient indicating the number of different combinations of \(n-1\) possible existing contenders on one of the \(k\) columns, taken \(c\) contenders at a time. Now, \(\Psi\) is given by:

\[
\Psi = \psi \sum_{0 \leq s \leq b} \pi_s(s) \left[1 - X^\beta_k(\beta, s)\right].
\]  

(5.5)

The expression for \(\Psi\) facilitates the determination of the state of the input port buffers. The input port queue can be modeled as a \((2\alpha+1)\)-state Markov chain (see Figure 5.7). The transition rates are determined by the offered load \(\rho\) and the crosspoint queue-set grant probability \(\Psi\). Let \(f\) be the probability that a cell has fanout 2 (a bipoint connection). In this model, a bipoint connection, which is a two-copying task in the cell recycling technique, is treated as two independent point-to-point connections. The upper row of the Markov chain depicts the state of the queue when a cell needs no additional copy. However, if the cell has fanout 2, once the operation for its first copy in the lower row of the chain is completed, it proceeds to the second copy of the cell on the upper row. The state of the queue can be changed from 0 to 1 with transition probability \(\rho f\) if the cell has fanout 1; or it can enter state 1' with transition probability \(\rho f\) if the cell has fanout 2. Upon the arrival of a fanout-2 cell at the head of queue, a state in the upper chain is changed to a lower one with probability of \(\rho f\Psi\) if a new cell
arrives at the queue, or $\bar{\rho} f \Psi$ if no new cell arrives. Similarly, any of the lower chain states can be changed with probabilities $\rho \Psi$ or $\bar{\rho} \Psi$ to an upper one as soon as the first copy of a fanout-2-cell is completed. These transition probabilities are free of $f$ since only the first copy has been completed at this point. The second copy is processed in the upper chain independently. The probability that an input port buffer contains exactly $s$ cells is computed recursively by:

$$\pi_i(0) = \pi_i(0)\bar{\rho} + \pi_i(1)\rho \Psi,$$

$$\pi_i(1) = \pi_i(0)\bar{\rho} f + \pi_i(1)(\bar{\rho} \Psi + \rho \bar{\Psi} + \rho f \Psi) + \pi_i(1')\bar{\rho} \Psi + \pi_i(2)\rho \bar{\Psi},$$

$$\pi_i(1') = \pi_i(0)\rho f + \pi_i(1)\rho f \Psi + \pi_i(1')\bar{\rho} \Psi + \pi_i(2)\rho \bar{\Psi},$$

$$\pi_i(2) = \pi_i(1)\rho \bar{\Psi} + \pi_i(1')\rho \Psi + \pi_i(2)(\bar{\rho} \Psi + \rho \bar{\Psi}) + \pi_i(2')\rho \Psi + \pi_i(3)\rho \bar{\Psi},$$

$$\pi_i(2') = \pi_i(1')\rho \Psi + \pi_i(2)\rho f \Psi + \pi_i(2')\rho \bar{\Psi} + \pi_i(3)\rho \bar{\Psi},$$

$$\vdots$$

$$\pi_i(s) = \pi_i(s - 1)\rho \bar{\Psi} + \pi_i(s' - 1)\rho \Psi + \pi_i(s)(\bar{\rho} \Psi + \rho \bar{\Psi}) + \pi_i(s')\rho \Psi + \pi_i(s + 1)\rho \bar{\Psi},$$

$$\pi_i(s') = \pi_i(s' - 1)\rho \bar{\Psi} + \pi_i(s)\rho f \Psi + \pi_i(s')\rho \Psi + \pi_i(s + 1)\rho \bar{\Psi},$$

$$\vdots$$

$$\pi_i(\alpha) = \pi_i(\alpha - 1)\rho \bar{\Psi} + \pi_i(\alpha' - 1)\rho \Psi + \pi_i(\alpha)(\bar{\Psi} - \rho f \Psi) + \pi_i(\alpha')\rho \Psi$$
\[ \pi_i(\alpha') = \pi_i(\alpha' - 1)p\Psi + \pi_i(\alpha')(\rho + p\Psi) + \pi_i(\alpha)\rho f\Psi. \tag{5.14} \]

Next, let \( w_k^\beta(r, s) \) be the number of distributions that leave exactly \( r \) buffers of capacity \( \beta \) "not full" while a queue-set contains \( s \) cells, and let \( W_k^\beta(r, s) \) be the probability that exactly \( r \) buffers are not full when a queue-set contains \( s \) cells, therefore:

\[ w_k^\beta(r, s) = \binom{k}{r} z_r^{\beta-1}(s - (k - r)\beta), \tag{5.15} \]

and then,

\[ W_k^\beta(r, s) = \frac{w_k^\beta(r, s)}{z_k^\beta(s)}. \tag{5.16} \]

Now, we need to compute the probability that exactly \( j \) cells enter a queue-set when the queue is in state \( s \) at the beginning of the current cell cycle. The probability that a cell is available to enter a particular buffer of a queue-set, \( a_x \), is calculated based on \( a \) that was introduced earlier, and is given by:

\[ a_x = \frac{1}{n} \left[ 1 - \left( 1 - \frac{a}{k} \right)^n \right]. \tag{5.17} \]

Then \( p(j, s) \) is:

\[ p(j, s) \approx \sum_{j \leq r < k} W_k^\beta(r, s) \binom{r}{j} (a_x)^j (1 - a_x)^{r-j}, \tag{5.18} \]

We also need to compute the probability that exactly \( j \) cells leave a queue-set when the queue is in state \( s \) at the beginning of the current cell cycle. For \( q(j, s) \), it should be remembered that, due to the queueing model, there is only one output for every queue-set, so the probability is introduced by a simple form of:

\[ q(j, s) = \begin{cases} 
0 & \text{if } (s = 0 \land j = 1) \lor (s = 1 \land j = 0) \lor (j > 1) \\
1 & \text{if } (s \geq 1 \land j = 1) \lor (s = 0 \land j = 0). 
\end{cases} \tag{5.19} \]

Each queue-set is modeled as a \((b + 1)\)-state Markov chain \((b = k\beta)\). The transition probability, \( \lambda(\hat{s}, s) \), and the next state probability, \( \pi_x(s) \), defined earlier can be determined as follows:

\[ \lambda(\hat{s}, s) = \min\{k, h-s\} \sum_{h=\max\{0, s-\hat{s}\}} p(h, \hat{s})q(h - (s - \hat{s}), \hat{s}), \tag{5.20} \]
where \(0 \leq (s' \text{ and } \delta) \leq b\), and:

\[
\pi_x(s) = \min_{\delta = \max\{0, s-k\}} \sum_{\delta' = \max\{0, s-k\}} \pi_x(\delta') \lambda(\delta', s).
\]  

(5.21)

5.2.2. Throughput and Delay

By putting together all of the above parameters, the system throughput \((T)\) and delay \((D)\) can be computed. The definition of throughput is similar to the one made in Chapter 3 where it refers here to the number of cells leaving a queue-set per output link per cell cycle. Since the entire network is modeled as one middle stage queue, the throughput, on the other hand, is the probability of having at least one cell in a queue-set and is simply:

\[
T = 1 - \pi_x(0).
\]  

(5.22)

Due to the definition given by Equation (5.19), clearly \(q(j, s) \in \{0, 1\}\). Thus, so long as there is at least one cell in the queue-set, \(q(j, s) = 1\), and \(T\) can be expressed free of \(q(j, s)\). The delay is defined to be the number of cell cycles that a given cell spends in the crossbar from the time it enters the network until it exits the network. This is another interpretation of the ratio of queue length and the arrival rate. Let the queue length of the network and the input port processors be denoted by \(Q_x\) and \(Q_i\), respectively. They are expressed by:

\[
Q_x = \sum_{0 \leq s \leq b} s \pi_x(s);
\]  

(5.23)

and

\[
Q_i = \sum_{0 \leq j \leq a} j \pi_i(j).
\]  

(5.24)
The arrival rate at the input port buffers is the offered load, $\rho$, and the arrival rate for each queue-set can be defined by $A$:

$$A = \sum_{0 \leq j \leq k} \sum_{0 \leq s \leq b} \pi_x(s)jp(j, s).$$  \hspace{1cm} (5.25)

Thus, the total delay is the summation of the delays:

$$D = \frac{Q_x}{A} + \frac{Q_i}{\rho}.$$  \hspace{1cm} (5.26)

5.3. Performance Evaluation Results

In this section, the results of the performance analysis are presented and discussed. There are three main variables on which the system throughput ($T$) and delay ($D$) measurements are based. These three variables are: offered traffic load ($\rho$), number of input/output ports ($n$), and number of columns ($k$). A program was written in C++ to compute the performance metrics.

5.3.1. Effect of Increasing Offered Load

Figures 5.8, 5.9, 5.10, and 5.11 show the first set of results. In Figures 5.8 and 5.9, we examine variations of incoming traffic load on a 16-port MBC network. In these examples, $\beta = 1$, and the fraction of multipoint cells is assumed to be $f = 0.75$. The results of the network throughput and delay are plotted where the number of columns takes three values of 16, 24, 32. As expected, the throughput is improved when $k$ increases. This is mainly due to availability of larger number of columns for cells waiting in the input buffers.

In general, the throughput curves demonstrate a different behavior compared to typical multistage switch throughputs. An obvious difference is that the throughput of the MBC never flattens out. Accordingly, the delay curves show that the average time
Figure 5.8: Throughput ($T$) of a 16-port network when offered load ($\rho$) and number of columns ($k$) vary.

Figure 5.9: Delay ($D$) in a 16-port network when offered load ($\rho$) and number of columns ($k$) vary.
Figure 5.10: Throughput ($T$) of a 64-port network when offered load ($\rho$) and number of columns ($k$) vary.

Figure 5.11: Delay ($D$) in a 64-port network when offered load ($\rho$) and number of columns ($k$) vary.
that cells spend in the network can not exceed 2.2 cell times. This amount of time was predictable since cells pass through only two stages of switching from an input port to their destinations. This behavior explains the better performance capability of an MBC network compared to a typical multistage switch. We remember that these performance benefits were not made at a additional cost, but as we evaluated the potentials of the wafer-scale technology, the architectural advantages are automatically acquired with the use of the WSI.

Figures 5.10 and 5.11 show the performance results with the same conditions as for the previous set of curves but for 64-port networks. For this size network, the number columns takes three values of 64, 96 and 128. The plots of both throughput and delay appear to be slightly different compared to the results for 16-port networks. This behavior also describes the stability of the MBC against enlarging the number of input/output ports.

5.3.2. Effect of Increasing Number of Ports

The next set of results is shown in Figures 5.12 and 5.13. The curves give the numerical results on the system performance with various network dimensions. This set of results is of particular interest since it provides the performance of networks for various size wafers. These figures plot the delay and throughput when the network dimension and the number of columns simultaneously increase. As discussed in the previous section, the MBC network demonstrate a stable performance when the network dimensions (both of rows and columns) increases.
Figure 5.12: Throughput ($T$) when the network dimension ($n$) and the number of columns ($k$) simultaneously increase.

Figure 5.13: Delay ($D$) when the network dimension ($n$) and the number of columns ($k$) simultaneously increase.
Figure 5.14: Throughput ($T$) versus the number of columns ($k$) where the number of buffers per crosspoint ($\beta$) is 1 or 2.

Figure 5.15: Average network delay ($D$) versus the number of columns ($k$) where the number of buffers per crosspoint ($\beta$) is 1 or 2.
5.3.3. Effect of Increasing Number of Columns

Next, we analyze the effect of increasing the number of columns for a network with a fixed number of inputs/outputs. Results are shown in Figures 5.14 and Figure 5.15. Although, a large number of buffer slots at each node seems to be impractical, as is seen in these figures, we examine the system performance for the two values of $\beta = 1$ and $\beta = 2$. The results, as expected, explicitly imply the improvement of the system throughput when the number of columns increases. It can be seen, however, that the higher number of buffer slots has a small impact on the system throughput and delay.

5.4. Evaluation of Complexity

The complexity of the switch fabric for fabrication on a single wafer using a multipath buffered crossbar consisting of $n$ input ports, $n$ output ports, and $k$ columns is denoted by $C_{net}$ and is equal to:

$$C_{net}(\beta, k, n) = k \cdot n \left[ C_b(\beta) + C_c + 2C_s(\beta) \right] + n \left[ C_{ipp} + C_{opp} \right], \quad (5.27)$$
where \( C_b(\beta) \), \( C_c \), and \( C_s(\beta) \) are the complexities of the buffers, the controller, and the switches at each crosspoint, respectively and \( C_{ipp} \) and \( C_{opp} \) indicate the complexities of the IPP and OPP, respectively. Then \( C_{net} \) per port is given by:

\[
C_{net}(\beta, k)/n = k \left[ C_b(\beta) + C_c + 2C_s(\beta) \right] + [C_{ipp} + C_{opp}].
\]

With \( \beta = 1 \) and \( k = n = 256 \), and assuming \( C_b(\beta) = 2.85K, C_c = 790, C_s(\beta) = 216 \), and \( C_{ipp} + C_{opp} \approx 3.2M \) transistors, the total transistor cost per port would be estimated to be \( C_{net}/n \approx 4.24M \) transistors. Note that by not including the complexities of the IPP and OPP, \( C_{net}/n \) will be reduced significantly. One way to enhance the throughput of the system is to increase the number of columns, \( k \), or in a naive way increase the number of buffers (\( \beta \)) at each crosspoint. Clearly there is a trade-off between the throughput and the total complexity of the switch fabric which is a function of \( \beta \) and \( k \). Figure 5.16 gives a numerical result for the complexity of the switch fabric per port versus \( \beta \) and \( n \).

5.5. Estimation of Area

After evaluation of complexity, a basic concern here is the size restrictions placed on the proposed network by the wafer. We use the results of the complexity evaluation presented in Section 5.4 to estimate the area required for the fabrication of the MBC on a single wafer. According to the numerical results of this section, each crosspoint \( (C_b(\beta) + C_c + 2C_s(\beta)) \) consumes about 3.8K transistors for \( \beta = 1 \), and each pair of IPP and OPP require 3.2M transistors. We assume each transistor occupies \( 400\lambda^2 \), where \( \lambda = t/2 \) is one half the minimum feature size. By adding a small percentage of the total area of transistors for external routing, we can estimate the area of an entire MBC network for the fabrication on a single wafer. For example, when \( n = 256 \), we compute \((3,800 \times 256^2) + (3,200,000 \times 256)\) transistors. At 0.5\( \mu \)m feature size, we need 25\( \mu \)m\(^2\) per transistor or approximately 275 square centimeter for the entire switching system.
Figure 5.17: Evaluation of the network area on a single wafer versus the technology feature size ($t$).

Figure 5.17 depicts the results of this study for a 64-port and a 256-port network. In this figure we also give the maximum area available on 5-, 6-, and 8-inch wafers. For the estimation of the available area, we consider a large square at the center of a wafer and four narrow rectangles attached to each side of the square.

5.6. Evaluation of Yield for WSI Technology

The negative binomial distribution derived from Equation (4.4) is used to model the circuit integration yield of the MBC network. By substitution of the area required for each self-driven crosspoint, denoted by $A_{scd}$ in Equation (4.4), the overall yield for the MBC network is given by:

$$y = \frac{1}{\left[1 + \frac{A_{scd}D_0}{(\sigma/\mu)^2}\right](\sigma/\mu)^T}, \quad (5.29)$$

where the numerical results are shown in Table 5.1. For this model we use the practical lower limits obtained from the manufacturing statistics provided in [28] and [49] such as the mean numbers of defects, $D_0 \in \{1, 2, 3\}$, and the variance-to-squared-mean ratio,
Table 5.1: Yield ($y$) evaluation for the crosspoint and the port processors of the multipath buffered crossbar.

<table>
<thead>
<tr>
<th></th>
<th>$\beta$</th>
<th>$D_0 = 1$</th>
<th>$D_0 = 2$</th>
<th>$D_0 = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>crosspoint</td>
<td>1</td>
<td>0.9983</td>
<td>0.9967</td>
<td>0.9951</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.9971</td>
<td>0.9942</td>
<td>0.9913</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.9958</td>
<td>0.9917</td>
<td>0.9876</td>
</tr>
<tr>
<td>IPP/OPP</td>
<td></td>
<td>0.8811</td>
<td>0.7675</td>
<td>0.6927</td>
</tr>
</tbody>
</table>

$(\sigma/\mu)^2 \approx 0.7$. The results indicate how yield is affected when the area of each crosspoint increases by changing $\beta$.

5.7. Remarks

In the next chapter, we will be extending further the concept of large-scale and high-speed switches by introducing a switching system with a full regular structure using wafer-scale integration. This architecture is an alternative to the one presented in this chapter, and it has the advantage of more regularity in its structure. We will be showing by analysis and simulation that the performance of this alternative network is comparable to the performance of the MBC if the factor of speed advantage\(^2\) is taken into account.

---

\(^2\) the ratio of speed: internal links to external links
6. **A MANHATTAN STREET SWITCHING NETWORK**

This chapter\(^1\) proposes another high-speed ATM switch architecture which has lower complexity compared to the systems presented in the previous chapters. The network topology is based on the *Manhattan-street network* (MSN). The MSN has a *cyclic* (torus) structure and belongs to the class of deflection-routing networks. In this class of networks, if two cells arrive at time \(t\) from the neighboring nodes and both cells require the use of a given output to reach their destinations in the most direct way, a conflict may arise which can be resolved by temporarily deflecting one of the cells on an undesired link and allowing the other one to be forwarded to the desired link. We use the MSN topology for ATM communications applications and utilize shared buffering at each node. With shared buffers the deflection of a cell occurs when all buffer slots are full. This method potentially increases the system performance and reduces the number of deflections. We evaluate the effects of high offered traffic loads, large shared buffers, and increasing network size on the network performance.

This chapter is organized as follows, Section 6.1 briefly reviews the topology and properties of the MSN. In Section 6.2 the structure of a node with shared buffering and its potential effects on performance are discussed. Section 6.3 classifies types of MSN nodes based on the routing algorithm and presents an analysis for counting the number of nodes of each type. In Section 6.4 the performance of the network under deflection routing using a stochastic model for the queueing-deflection process is analyzed, and some numerical results are presented. Finally, in Sections 6.6, 6.7, and 6.8 some of the hardware and technological aspects including the complexity, yield, and area are evaluated.

\(^1\)This chapter together with Chapter 7 has been (or will be) published in part in [39], [40] [44].
Figure 6.1: A 4×4 Manhattan-street network.

6.1. Manhattan Street Network

The *Manhattan-street network* (MSN) has been proposed by Maxemchuk [31] [32] for implementation in geographically distributed computer networks. The MSN is a regular mesh network that can be realized as a collection of horizontal and vertical rings as shown in Figure 6.1. The links resemble the streets and avenues in Manhattan which alternate in direction. With the *torus*-shaped topology embedded in the network, at each node there are two links arriving and two links leaving regardless of the network size. Although it is not indicated in the figure, at each node an incoming link and an outgoing link connect the network to a local processor which generates and receives data sent through the network. Routing in the MSN relies on *deflection*. When two cells request the same outgoing link, only one of them is forwarded on the preferred link while the other one is deflected on the second link. By maintaining this rule in the system, once a cell is admitted to a network, it is not discarded if congestion occurs. Instead, the cell is misrouted temporarily but will ultimately reach its destination. However, deflection of cells onto longer paths causes additional delay.
6.1.1. Routing in the MSN

In [32], four distinct distributed routing rules for the Manhattan street network have been investigated. The rules take advantage of the regular structure of the network. Among these methods of routing, deterministic Rule 2 is chosen here as a routing strategy. We select this rule because it demonstrates a relatively high efficiency at relatively low complexity [32]. We now briefly review this rule and later apply it to both analysis and simulation.

Because of the cyclic structure of the MSN, any node can be considered to be in the center of the network as illustrated in Figure 6.2. When routing a cell, we treat the destination as though it were at location (0,0). This can be accomplished by a simple transformation on addresses. The transformation effectively renumbers and re-orient the network so that:

- The network is partitioned into four quadrants $Q_1$, $Q_2$, $Q_3$ and $Q_4$.
- The destination node is realized approximately in the center of the network with a relative address of (0,0).
• The destination node has both row and columns directed toward decreasing numbered nodes.

The boundaries of the above four regions in an $N \times N$ network are obtained from the following expressions:

\[
Q_1 = \{(i,j) : 1 \leq i \leq N/2, \ 1 \leq j \leq N/2\}, \quad (6.1)
\]

\[
Q_2 = \{(i,j) : 1 \leq i \leq N/2, \ -(N/2 - 1) \leq j \leq 0\}, \quad (6.2)
\]

\[
Q_3 = \{(i,j) : -(N/2 - 1) \leq i \leq 0, \ -(N/2 - 1) \leq j \leq 0\}, \quad (6.3)
\]

\[
Q_4 = \{(i,j) : -(N/2 - 1) \leq i \leq 0, \ 1 \leq j \leq N/2\}. \quad (6.4)
\]

In each quadrant there are fixed, preferred directions such that a cell approaches and is channeled to its destination by taking these directions. The routing rule uses the preferred directions to route each cell to its destination along a shortest path. In addition to these partitions, there are other subareas within quadrants $Q_2$ and $Q_4$ that have special rules. These are defined as follows:

\[
r_2 = \{(i,j) : 1 \leq i \leq N/2, \ j = 0\}, \quad (6.5)
\]

\[
c_2 = \{(i,j) : i = 1, \ -(N/2 - 1) \leq j \leq 1\}, \quad (6.6)
\]

\[
Q_2^- = Q_2 - (r_2 \cup c_2). \quad (6.7)
\]

and also for quadrant $Q_4$:

\[
r_4 = \{(i,j) : i = 0, \ 1 \leq j \leq N/2\}, \quad (6.8)
\]

\[
c_4 = \{(i,j) : -(N/2 - 1) \leq i \leq -1, \ j = 1\} \quad (6.9)
\]

\[
Q_4^- = Q_4 - (r_4 \cup c_4). \quad (6.10)
\]

The arrows in the regions $Q_1$, $Q_2^-$, $Q_3$, $Q_4^-$, $r_2$, $c_2$, $r_4$, and $c_4$ define the preferred directions for those regions. A cell at a node in a given region is routed in the preferred
direction if exactly one output is oriented in a preferred direction. If neither or both outputs lead in a preferred direction, the cell is routed arbitrarily to one of the two outputs.

6.2. Node Structure with Shared Buffering

In a deflection-routing network, if two cells from the neighboring nodes arrive at the same time and both cells require use of a given output, the conflict is resolved by temporarily deflecting one of the cells on an undesired link and allowing the other one to be forwarded to the desired link. However, deflection routing by itself as a contention resolution strategy can waste bandwidth within the network by requiring cells to follow suboptimal routes. If a node has no internal storage, deflection is necessary to avoid blocking or discarding the cell. It is apparent that augmenting the Manhattan street networks with buffers at every node potentially offers better performance. In this context, [11] and [12] have examined the effect of output buffering in the Manhattan-street network. We extend this work and combine the two output buffers of each node, letting the inputs and outputs share an internal queue to reduce the frequency of the full buffer condition and hence reducing the number of deflections further.

Before discussing the effect of the above modification in depth, we compare the two buffering methods. Assume a MSN with output buffering where the number of slots of both buffers is $\beta$. An incoming cell may encounter two different cases. If the desired buffer is not full, the cell is queued in the buffer and is not deflected. Even if a second incoming cell from the other input is destined for the same buffer, both cells are stored in the desired buffer as long as the buffer has at least two empty slots. A second case that might occur is when a cell faces a full buffer and is forced to enter the other buffer. With this strategy a cell must be routed to one of the two buffers upon its arrival. If
the cell is deflected to the wrong buffer this decision can not be changed at a later time if the desired buffer allows admission of new arrivals.

By combining the two separate buffers into a shared buffer with capacity of $b = 2\beta$ both the inputs are allowed to access all buffer slots. With this buffering structure cells are stored in the shared (between inputs and outputs) buffer slots regardless of their destinations, and deflection occurs only when all buffer slots are occupied. The structure of a node in a two-connected network is shown in Figure 6.3. Note that the routing within the network is implemented by $2 \times 2$ nodes, but as mentioned before, there is one incoming external link and one outgoing external link joining each node to a local processor. In the ATM context, this processor would implement the interface to an external transmission link. Thus, each node is in fact a $3 \times 3$ switch. With the shared buffer structure, an incoming cell is immediately stored in one of the $b$ buffer slots. A $3 \times b$ crossbar implements the switching functions. At each cell cycle, a cell competes to access the desired outgoing link based on a priority field, and the winning cell is allowed to exit through a $b \times 3$ crossbar.

The method of shared buffering in the MSN would potentially reduce the number of deflections compared to the similar but nonbuffered networks or MSNs with different
buffering strategies. In [6] and [61] comprehensive studies on the performance of the shared buffering strategy for switching networks have been made. The results of this study show that switching systems with shared buffers at each switch element provide better overall performance.

6.2.1. Description of Node Operation

When cells arrive at a node, the node determines how they should be routed. There are four possibilities; type-1 cells are transmitted to the local processor (through the external link), type-2 cells are addressed to the row output (internal link), type-3 cells are addressed to the column output (internal link), and finally type-4 cells can be sent to either the row or column. The last possibility arises because the grid structure of the MSN provides a multiplicity of equally good paths between most source-destination pairs. The routing decision for a cell is made when the cell arrives and is stored in the control circuitry of the node.

The MSN provides no flow control between adjacent nodes in the network, but does provide flow control back to the local processor. This means that the node must be prepared to receive two cells each cycle. To ensure that two cells can be received and stored in one cycle, the output control always transmits enough cells to ensure that two buffer slots are available for arriving cells. If at the start of a node operational cycle, the shared buffer is full and all stored cells are of the same type and none is of type 4, one of the stored cells is selected for deflection and sent to a row or column output that would not otherwise be used.
6.2.2. Port Processors

At the incoming external link of a node, an input port processor (IPP) is used to convert the ATM cell to the local network format. The IPPs have lookup tables to determine the routing information for cells. Due to the limited capacity of the network, an input buffer, as shown in Figure 6.3 accepts new arrivals if there are empty slots; otherwise the cells are discarded. Once a cell is admitted to the network, due to the queueing-routing strategy explained earlier no cells are lost. At the output side, output port processors (OPPs) accept cells from the MSN and resequence them. A resequencing operation is needed for misordered cells. This misordering is caused by the multiplicity of paths and the resulting variation in delays in the queues throughout the network.

6.3. Number of Nodes of Specific Type

In Section 6.1.1, the routing algorithm for the MSN was explained. We noticed that in each of the eight areas or subareas, there are a number of nodes in which a cell can take either outgoing link to reach to its destination using a shortest possible path. This is a consequence of the topological structure of the MSN which provides a multiplicity of shortest paths between most pairs of nodes. Here, these types of nodes are referred to as don't-care nodes. We use the percentage of nodes in an MSN that are don't-care nodes to estimate the probability that a cell entering an arbitrary node is of type 4. In this section we determine the number of don't-care nodes in an MSN of arbitrary size.

In an $N \times N$ network, let $\nu_{Q_1}$, $\nu_{Q_2}$, $\nu_{Q_3}$, and $\nu_{Q_4}$ be the number of don't-care nodes in quadrants $Q_1$, $Q_2$, $Q_3$ and $Q_4$ respectively. In quadrant $Q_1$, nodes (1,1), (1,3), ..., are don't-care, as are nodes (2,2), (2,4), (2,6), ... . In general, all nodes $(i,j)$ in $Q_1$ for which $i+j$ are even numbers are don't-care nodes. This observation leads to the result:
\[
\nu_{Q_1} = \begin{cases} 
\frac{1}{2} \left( \frac{N}{2} \right)^2 & \text{if } \frac{N}{2} \text{ is even} \\
\frac{1}{2} \left[ \left( \frac{N}{2} \right)^2 + 1 \right] & \text{if } \frac{N}{2} \text{ is odd}
\end{cases}
\]

We can combine the results of the two cases in one formula:

\[
\nu_{Q_1} = \left\lfloor \frac{1}{2} \left( \frac{N}{2} \right)^2 \right\rfloor.
\]  \hspace{1cm} (6.11)

For the quadrant \( Q_2 \), we need to consider two other subareas, \( r_2 \) and \( c_2 \), illustrated in Figure 6.2. Note that according to the property of the addressing scheme with respect to the destination at (0,0), no "don't-care node" exists in either of subareas \( r_2 \) or \( c_2 \). Hence:

\[
\nu_{Q_2} = \begin{cases} 
\frac{1}{2} \left[ \left( \frac{N}{2} - 1 \right)^2 + 1 \right] & \text{if } \frac{N}{2} \text{ is even} \\
\frac{1}{2} \left( \frac{N}{2} - 1 \right)^2 & \text{if } \frac{N}{2} \text{ is odd}
\end{cases}
\]

Similarly, we can combine the results of the above two cases in one formula as follows:

\[
\nu_{Q_2} = \left\lfloor \frac{1}{2} \left( \frac{N}{2} - 1 \right)^2 \right\rfloor.
\]  \hspace{1cm} (6.12)

The number of don't-care nodes in quadrant \( Q_3 \) is clearly the same as the number in quadrant \( Q_1 \) minus one. Notice that the destination node is always in quadrant \( Q_3 \) and thus it should not be taken into account.

\[
\nu_{Q_3} = \left\lfloor \frac{1}{2} \left( \frac{N}{2} \right)^2 \right\rfloor - 1.
\]  \hspace{1cm} (6.13)

Finally, from inspection of Figure 6.2, \( \nu_{Q_4} \) is the same as \( \nu_{Q_2} \), so:

\[
\nu_{Q_4} = \left\lfloor \frac{1}{2} \left( \frac{N}{2} - 1 \right)^2 \right\rfloor.
\]  \hspace{1cm} (6.14)
Table 6.1: Examples on the number of don't-care nodes for different network sizes.

<table>
<thead>
<tr>
<th>network size</th>
<th>( \nu_{Q_1} )</th>
<th>( \nu_{Q_2} )</th>
<th>( \nu_{Q_3} )</th>
<th>( \nu_{Q_4} )</th>
<th>( \nu_{sum} )</th>
<th>( \nu )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n = 16 )</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>53.3%</td>
</tr>
<tr>
<td>( n = 64 )</td>
<td>8</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>25</td>
<td>39.6%</td>
</tr>
<tr>
<td>( n = 1296 )</td>
<td>162</td>
<td>145</td>
<td>161</td>
<td>145</td>
<td>1213</td>
<td>47.3%</td>
</tr>
<tr>
<td>( n = 2500 )</td>
<td>313</td>
<td>288</td>
<td>312</td>
<td>288</td>
<td>1201</td>
<td>48.0%</td>
</tr>
</tbody>
</table>

Let \( \nu_{sum} \) be the total number of don't-care nodes in the MSN. Given \( N \), we can make a summation over the above expressions to compute \( \nu_{sum} \):

\[
\nu_{sum} = \sum_{i \in \{1,2,3,4\}} \nu_{Q_i} = 2 \left[ \frac{1}{2} \left( \frac{N}{2} \right)^2 \right] + 2 \left[ \frac{1}{2} \left( \frac{N}{2} - 1 \right)^2 \right] - 1
\]

\[
= \left( \frac{N}{2} \right)^2 + \left( \frac{N}{2} - 1 \right)^2
\]

\[
= \frac{1}{2} N^2 - N + 1.
\] (6.15)

Let \( \nu \) be the overall percentage of don't-care nodes among the other nodes. Since the total number of nodes in the network not including the destination is \( N^2 - 1 \), then:

\[
\nu = \frac{1}{N^2 - 1} \cdot \nu_{sum} \cdot 100\%.
\] (6.16)

In Table 6.1, we show some numerical examples of the above set of formulas to see how quickly \( \nu \) converges. Clearly \( \nu \to 50\% \) as \( N \) gets large. As mentioned above, we use the percentage of don't-care nodes to estimate the number of type-4 cells. We examine how this affects the accuracy of the performance analysis in Section 7.2.1 by comparison with simulation results.

6.4. Performance Analysis

A number of papers have been published to characterize the performance of networks using deflection routing including [20] [32] [33]. The previous results revealed that
the use of buffers in the MSN can reduce the number of misroutings and therefore can potentially enhance the performance. In this section, an analytic model for the Manhattan-street network is introduced. We examine the effect of shared buffering on the network performance. For the performance analysis the following variables are defined:

\[ n: \quad \text{Total number of nodes in the network;} \]
\[ N: \quad \text{Number of nodes at each side of the network } N = \sqrt{n}; \]
\[ \alpha: \quad \text{Number of slots in an input buffer;} \]
\[ b: \quad \text{Number of slots in a shared buffer;} \]
\[ j_k: \quad \text{Number of cells of type } k \in \{1, 2, 3, 4\}: \]
\[ \text{type-1: cells addressed to the local processor,} \]
\[ \text{type-2: cells addressed to a row output,} \]
\[ \text{type-3: cells addressed to a column output,} \]
\[ \text{type-4: cells that can be sent to either a row or a column output;} \]
\[ \rho: \quad \text{Offered load from the external link to the input buffer;} \]
\[ \rho_x: \quad \text{Offered load from the input buffer to the shared buffer;} \]
\[ \rho_i: \quad \text{Offered load from internal link } i \text{ to the shared buffer, } i \in \{1, 2\}; \]
\[ \rho_{xk}: \quad \text{Offered load from the input buffer containing cells of type } k, \]
\[ k \in \{2, 3, 4\}; \]
\[ \rho_{ik}: \quad \text{Offered load from internal link } i \text{ containing cells of type } k, i \in \{1, 2\}, \]
\[ k \in \{1, 2, 3, 4\}; \]
\[ \pi_x(j): \quad \text{Probability that the input buffer at a given node contains} \]
\[ \text{exactly } j \text{ cells;} \]
\[ \pi(j): \quad \text{Probability that the shared buffer at a given node contains } j_1, j_2, j_3, \text{ and } j_4 \text{ cells of type } 1, 2, 3, \text{ and } 4 \text{ respectively where } j \]
is a state vector $[j_1, j_2, j_3, j_4]$;

$\varphi$: Probability that a node can accept a new cell from its local input buffer;

$p(j)$: Probability that $j$ cells of type 1, 2, 3, and 4 enter a node in a given cell cycle where $j$ is a vector $[j_1, j_2, j_3, j_4]$;

$\ell_s$: Average length of shortest path between a pair of nodes in the MSN;

$\ell_d$: Average number of extra hops taken by cells as a result of deflections;

$\Delta$: Average number of deflections;

$\delta$: Probability that a cell gets deflected;

$\nu$: Overall percentage of don't-care nodes among the other nodes.

We also define $\Pi(j, \text{condition})$ to be:

$$
\Pi(j, \text{condition}) = \sum_{j, j_1 + j_2 + j_3 + j_4, \text{condition}} \pi(j). \quad (6.17)
$$

For example, $\Pi(3, j_1 = 0)$ is the probability that the buffer has 3 cells, none of which is of type 1. We often omit the condition, in which case $\Pi(j)$ just represents the probability that a given node has exactly $j$ cells.

The state of the input buffer is explicitly modeled by a discrete time birth-death process. According to the operational description of a node given earlier, the input buffer is granted permission to deliver cells to the shared buffer so long as the number of cells in the shared buffer is less than $b - 1$. This mechanism is carried out by a grant flow control as shown in Figure 6.3, and the probability that a grant is given is denoted by $\varphi$. Thus, $\varphi$ depends on the state of the shared buffer and can be expressed as:

$$
\varphi = 1 - (\Pi(b - 1) + \Pi(b)). \quad (6.18)
$$

Using $\varphi$ and the external offered load $\rho$, all states of the input queue can be determined from the transition probabilities of the Markov chain describing the input buffer as
follows:

\[
\pi_x(0) = \pi_x(0)\varphi + \pi_x(1)\varphi, \quad (6.19)
\]

\[
\pi_x(1) = \pi_x(0)\rho + \pi_x(1)(\rho\varphi + \rho\varphi) + \pi_x(2)\varphi, \quad (6.20)
\]

\[\vdots\]

\[
\pi_x(s) = \pi_x(s-1)\rho\varphi + \pi_x(s)(\rho\varphi + \rho\varphi) + \pi_x(s+1)\varphi, \quad (6.21)
\]

\[\vdots\]

\[
\pi_x(\alpha-1) = \pi_x(\alpha-2)\rho\varphi + \pi_x(\alpha-1)(\rho\varphi + \rho\varphi) + \pi_x(\alpha)\varphi, \quad (6.22)
\]

\[
\pi_x(\alpha) = \pi_x(\alpha-1)\rho\varphi + \pi_x(\alpha)\varphi. \quad (6.23)
\]

where \( \alpha \) is the number of input buffer slots. Next, let \( \rho_x \) be the probability that a cell is available to enter the shared buffer from the input buffer. This term is calculated from the state of the input queue when it is nonempty:

\[
\rho_x = 1 - \pi_x(0). \quad (6.24)
\]

As defined above, let \( \rho_1 \) (\( \rho_2 \)) be the probability that the row (column) predecessor of a given node has a cell for it. By the symmetry of the network and the uniform traffic assumption, \( \rho_1 \) (\( \rho_2 \)) is also the probability that a node sends a cell on its row (column) output. Furthermore \( \rho_1 = \rho_2 \), thus:

\[
\rho_1 = \rho_2 = \sum_{j \in X} \pi(j) + \frac{1}{2} \sum_{j \in Y} \pi(j). \quad (6.25)
\]

where \( X \) consists of all state vectors \([j_1, j_2, j_3, j_4]\) that satisfy the following:

\[ [j_2 > 0] \]

\[ \vee [j_2 = 0 \land j_3 = 0 \land j_4 > 1] \]

\[ \vee [j_2 = 0 \land j_3 > 0 \land j_4 > 0] \]

\[ \vee [j_1 = 0 \land j_2 = 0 \land j_3 = b \land j_4 = 0] \]
and $Y$ consists of all state vectors $[j_1, j_2, j_3, j_4]$ that satisfy the following:

$$[j_2 = j_3 = 0]$$

$$\land [j_1 + j_4 = b]$$

$$\land [j_4 \leq 1]$$

Also, with term $\pi(j)$ the condition under which a cell must be deflected can explicitly be determined. Let $\delta$ be the probability that a node deflects a cell in a given cell cycle. A cell is deflected when all of the shared buffer slots are occupied by cells of type 1 or 2 or 3. This statement is expressed by:

$$\delta = \frac{1}{2} \rho_i \sum_{j \in Z} \pi(j),$$

(6.26)

where $Z$ consists of all state vectors $[j_1, j_2, j_3, j_4]$ that satisfy the following:

$$[j_1 = b] \lor [j_2 = b] \lor [j_3 = b].$$

Now, in order to compute the probability that a given number of cells enter a node, we first need the traffic load for cells of each specific type. Let $\rho_{x2}$, $\rho_{x3}$, and $\rho_{x4}$ be the offered loads from the input buffer for cells of type 2, 3, and 4 respectively. Notice here that, no cells of type 1 arrive from the external link as indicated in Figure 6.4.

The use of a uniform traffic model leads to balanced cell arrivals of type 2 and 3 from each internal link and thus $\rho_{x2} = \rho_{x3}$. In Section 6.3 we computed the percentage $\nu$ of nodes in a network that are don't-care nodes. We take the percentage of cells that are of type 4 to be equal to $\nu$. This gives

$$\rho_{x2} = \rho_{x3} = \frac{1}{2} (1 - \nu) \rho_x,$$

(6.27)

$$\rho_{x4} = \nu \rho_x.$$

(6.28)

Similarly for $i \in \{1, 2\}$, let $\rho_{i1}$, $\rho_{i2}$, $\rho_{i3}$, and $\rho_{i4}$ be the internal traffic arriving from an internal link that contain only cells of type 1, 2, 3, and 4 respectively. Relying on the
fact that no cells are lost within the network, the exiting load on the external link (the predecessor link of the OPP) should be equal to $\rho_x$. This load is clearly the result of the cell transmission from the two internal links. Therefore each internal link has an equal contribution of $\rho_x/2$ to the exiting traffic from the external link as shown in Figure 6.4. The internal traffic loads can be written as:

$$\rho_{i1} = \frac{1}{2} \rho_x,$$

$$\rho_{i2} = \rho_{i3} = \frac{1}{2} (1 - \nu) (\rho_i - \frac{1}{2} \rho_x),$$

$$\rho_{i4} = \nu (\rho_i - \frac{1}{2} \rho_x).$$  \hspace{0.5cm} \text{(6.29, 6.30, 6.31)}$$

Note that $\rho_i$ can be computed by $\rho_i = \ell \rho_x/2$ where $\ell$ is the average length of a path taken by a cell. Now, we can compute the probability that a given number of cells of various types enter a node. Assuming that an arriving cell from an external link never exits from the same node, we define the generating function $g(x, y, z)$:

$$g(x, y, z) = \frac{((1 - \rho_x) + (\rho_{x1} + \rho_{x3})y + \rho_{x4}z)}{((1 - \rho_1) + \rho_{11} x + (\rho_{12} + \rho_{13})y + \rho_{14} z)^2}. \hspace{0.5cm} \text{(6.32)}$$
This expression can be expanded into a polynomial on \( x, y, \) and \( z \). In this form, the coefficient of \( x^{j_1} y^{j_2} z^{j_3} \) represents the probability that exactly \( j_1 \) cells of type 1, \( j_2 + j_3 \) cells of types 2 and 3, and \( j_4 \) cells of type 4 enter in a single cycle. We call this probability \( p(j_1, j_2, j_3, j_4) \), or \( p(j) \) where \( j \) is a vector \([j_1, j_2, j_3, j_4]\). Notice that this analysis relies on the assumption that the types of arriving cells are independent.

We model the shared buffer by a Markov process. We define \( \lambda(j_1, j_2) \) to be the probability that if a shared buffer is in state \( j_1 \) in the current cycle, it will be in state \( j_2 \) during the next cycle. Let \( Q(j) \) be the vector \( k = [k_1, k_2, k_3, k_4] \) that represents the number of departing cells of each type when the state is \( j \). The transition probability is then computed by:

\[
\lambda(j_1, j_2) = \sum_{j_1 + h - Q(j_1) = j_2} p(h).
\]

The steady state probability for the shared buffer can then be determined from the balance equations:

\[
\pi(j_2) = \sum_{j_1} \lambda(j_1, j_2) \pi(j_1).
\]

We now take into account the impact of routing where a cell takes an average of \( \ell \) hops from its source to its destination. Let \( \ell_s \) be the average length of a shortest path from a source to destination assuming no deflections. Assuming a destination at \((0,0)\) and summing over all possible source addresses \((i, j)\) we can calculate the average distance as follows:

\[
\ell_s = \frac{1}{N^2 - 1} \sum_{i=-(N/2-1)}^{N/2} \sum_{j=-(N/2-1)}^{N/2} (|i| + |j| + \varepsilon),
\]

where the value of \( \varepsilon \) can be found in Table 6.2. It can be shown that this is equal to:

\[
\ell_s = \frac{1}{N^2 - 1} \left( \frac{N^3}{2} + \frac{N^2}{4} - N + 4 \left[ \frac{N}{4} \right]^2 + 4 \left[ \frac{N}{4} \right]^2 \right).
\]
Table 6.2: The exact values for the number of additional hops ($\epsilon$) a cell takes as a result of the fact that streets are always one way in all regions of the MSN.

<table>
<thead>
<tr>
<th>quadrant .</th>
<th>i even j even</th>
<th>i even j odd</th>
<th>i odd j even</th>
<th>i odd j odd</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_1$: $i &gt; 0, j &gt; 0$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>$Q_2$: $i \leq 0, j &gt; 0$</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$Q_3$: $i \leq 0, j \leq 0$</td>
<td>4*</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$Q_4$: $i &gt; 0, j \leq 0$</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

*2 if $N/2$ is odd and $|i| = |j| = N/2$.

To complete the analysis we need to determine the impact of deflection on the length of paths taken by cells. Thus, let $\ell_d$ be the average number of hops taken by cells as a result of deflections. Then, $\ell_d$ is:

$$\ell_d = g \cdot \Delta,$$  \hspace{1cm} (6.37)

where $\Delta$ is the average number of deflections in a given routing, and $g$ is the number of additional hops a cell takes per deflection. Examining the Manhattan street network, each deflection can cost a cell at most four additional steps, hence $g \leq 4$. For computing $g$, we need to calculate the sum of all additional hops cells take including all nodes and average it over the total number of nodes. In this case, one can see that don't-care nodes in which cells never get deflected must be excluded from the calculation. The number of don't-care nodes can be obtained from Equation (6.15). The exact value for $g$ can be given by:

$$g = \begin{cases} 
\frac{4(N^2-4)}{N^2+2N-4} & \text{if } \frac{N}{2} \text{ is even} \\
\frac{2(N^2+6N-16)}{N^2+2N-4} & \text{if } \frac{N}{2} \text{ is odd}.
\end{cases}$$

Next, $\Delta$ can be expressed in terms of $\delta$, the probability that a cell gets deflected which was given in Equation (6.26), therefore:

$$\Delta = \delta(\ell_{s} + \ell_{d}).$$ \hspace{1cm} (6.38)
By combining Equations (6.37) and (6.38), \( \ell_d \) can be expressed as follows:

\[
\ell_d = g\delta (\ell_s + \ell_d),
\]

(6.39)

hence:

\[
\ell_d = \left( \frac{g\delta}{1 - g\delta} \right) \ell_s.
\]

(6.40)

Recall \( \ell \) is the average shortest path length including deflections. The value of \( \ell \) is given by:

\[
\ell = \ell_s + \ell_d = \left( \frac{1}{1 - g\delta} \right) \ell_s.
\]

(6.41)

Given, the above equation, one can compute the steady-state probabilities for a representative node of the MSN using an iterative numerical computation. Given the steady-state probabilities, we can easily calculate the required performance metrics of interest. The throughput \( T \) can be extracted directly from the state of the shared buffers. According to the previous discussion, as long as there is one cell of type 1 in the buffer, one of the cells is guaranteed to exit from the node. This is basically due to the fact that there is no flow control mechanism at the node output. The throughput is then equal to:

\[
T = \sum_{j, j_i \neq 0} \pi(j).
\]

(6.42)

The average delay \( D \) is defined to be the number of cell cycles that a cell spends in an \( n \)-port network to reach its destination. Between the time a cell enters the network until it leaves, it passes through an average of \( \ell + 1 \) nodes. In the first stage of this queueing sequence, a cell stays in the input queue before entering the network. In the next \( \ell \) nodes, at each stage a cell is queued in the shared buffer. At each of these nodes, the cell is of type 2, 3, or 4. Finally, a cell stays in the shared buffer of the destination node where it is a type-1 cell. Let \( D_2 \) be the delay incurred by the input queue, and \( D_1, D_2, \)
$D_3$ and $D_4$ be the delays that a cell incurs in a shared queues where it is respectively of type 1, 2, 3, and 4. The total average delay can then be computed as follows:

$$D = D_x + \ell \left( \frac{1}{2}(1 - \nu)D_2 + \frac{1}{2}(1 - \nu)D_3 + \nu D_4 \right) + D_1.$$  \hfill (6.43)

Recall that $\nu$ represents the fraction of don’t-care nodes. In general, delay in a queue is computed by the ratio of the queue length and the arrival rate. Let the length of an input queue be denoted by $Q_x$. Let $Q_k$ be the number of cells of type $k$ in a shared queue, where $k \in \{1, 2, 3, 4\}$. We can define these two quantities by:

$$Q_x = \sum_{0 \leq j \leq \alpha} j \pi_x(j),$$  \hfill (6.44)

$$Q_k = \sum_{j \in J} j_k \pi_x(j).$$  \hfill (6.45)

The arrival rate for the input buffer is in fact the external offered load $\rho$. The arrival rate for the shared buffer depends on the type of cell and in a generic form is expressed by:

$$A_1 = \rho_x = 2\rho_{11},$$  \hfill (6.46)

$$A_k = \rho_{xk} + 2\rho_{1k}, \quad k \in \{2, 3, 4\}.$$  \hfill (6.47)

By Little’s law $D_x = Q_x/A_x$ and $D_i = Q_i/A_i$ for $i \in \{1, 2, 3, 4\}$. Substituting into Equation (6.43) gives the total average delay.

At this point, we briefly present a selected set of numerical results based on the analysis. For a complete presentation of the results and a comparison with simulation see Section 7.2 of the next chapter. We select two sets of plots shown in Figures 6.5 and 6.6 and a Table shown in Figure 6.3. Figure 6.5 shows the throughput with three sizes of shared buffers per port of each node ($B$).
Figure 6.5: Throughput ($T$) versus offered load ($\rho$) when the number of shared buffers per port of node ($B$) varies.

Figure 6.6: Delay ($D$) vs offered load ($\rho$) when the number of shared buffers per port of node ($B$) varies.
Table 6.3: The average number of deflections ($\Delta$) versus offered load ($\rho$) when the number of shared buffers per port of node ($B$) vary.

<table>
<thead>
<tr>
<th>offered load ($\rho$)</th>
<th>$B = 2$</th>
<th>$B = 3$</th>
<th>$B = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>0.2</td>
<td>0.00287</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>0.3</td>
<td>0.01492</td>
<td>0.00094</td>
<td>0.00000</td>
</tr>
<tr>
<td>0.4</td>
<td>0.06827</td>
<td>0.01342</td>
<td>0.00289</td>
</tr>
<tr>
<td>0.5</td>
<td>0.16270</td>
<td>0.05229</td>
<td>0.03031</td>
</tr>
<tr>
<td>0.6</td>
<td>0.27219</td>
<td>0.15430</td>
<td>0.10842</td>
</tr>
<tr>
<td>0.7</td>
<td>0.40320</td>
<td>0.28902</td>
<td>0.22082</td>
</tr>
<tr>
<td>0.8</td>
<td>0.56009</td>
<td>0.42964</td>
<td>0.36183</td>
</tr>
<tr>
<td>0.9</td>
<td>0.63034</td>
<td>0.49648</td>
<td>0.41461</td>
</tr>
<tr>
<td>1.0</td>
<td>0.65884</td>
<td>0.51038</td>
<td>0.42996</td>
</tr>
</tbody>
</table>

Table 6.3 provides numerical results using three different buffer sizes, $B = 1$, $B = 2$, and $B = 3$. When the offered load increases, as expected, the average number of deflections increases since with heavier traffic, a cell is deflected with a higher probability. Further increasing the load, however, causes the throughput and therefore the delay to fall slightly since the growing incoming traffic increases the number of deflections.

6.5. Example of Multicast Implementation

In this section, we give an example for cell multicasting in the Manhattan street network. Multicast communication is handled using the cell recycling technique introduced in [62] which implements an arbitrary multipoint connection by constructing a binary tree in which the leaves correspond to output ports of the switch. Cells are recycled through the switch fabric, copying by two on each successive pass.

In the Manhattan street network, a node can be used with different purposes. Figure 6.7 gives an example of a multicast tree for a cell with a fanout four. By the source node and destination node, we mean the terminals through which a cell is delivered and
Figure 6.7: A multicast tree.

exits respectively. A relay node refers to a location that determines the addresses of the next two copies. Finally, a copy node is a node through which a cell is replicated and forwarded to both outgoing links. In an \( N \times N \) network where \( N = \sqrt{n} \), the number of hierarchical levels of a multicast tree in which copy nodes are required relies on the global fanout of a cell \( F \in \{0, 1, ..., n - 1\} \). In a multicast tree, the number of levels containing at least one copy node is \( \lceil \log_2 F \rceil \). The number of relay nodes \( R \) is equal to \( F - 2 \).

The determination of physical locations of copy nodes and relay nodes with respect to cell destinations can be important. These locations will affect the capability of a network in handling high bandwidths. Assume in Figure 6.7 a cell at copy node \( c_1 \) (as a local source node) is to be replicated and sent to two final destinations \( d_1 \) and \( d_2 \) respectively. Let \( (i_1, j_1) \) and \( (i_2, j_2) \) be the addresses of two destination nodes \( d_1 \) and \( d_2 \), and \( (i_3, j_3) \) be the address of \( c_1 \). Nodes \( d_1 \) and \( d_2 \) can always be realized as two vertices of a rectangle in the MSN. The ideal location of a relay node \( r_1 \) is one of the immediate neighbors of \( c_1 \), \( (i_3 + 1, j_3) \) or \( (i_3, j_3 + 1) \), depending upon which one is a closer node to \( c_3 \). By maintaining this rule in the network, the length of a routing path between a copy node and a destination node is evaluated in the relay node at its earliest
Figure 6.8: The routing example of a cell with a fanout 4 in the Manhattan street network.
time and thus at its lowest waste of bandwidth. For the same reason as for relay nodes, the location of a copy node must be the closest node to the pair of destination nodes. In Figure 6.8 the copying steps of a fanout-4 cell are shown and the ideal locations of copy nodes and relay nodes are indicated.

6.6. Evaluation of Complexity

As the network dimension increases, the inclusion of a speed advantage is necessary to prevent the throughput from dropping. The required speed advantage raises the complexity as \( N \) grows since the width of the inter-node data path needs to be increased. Assuming an \( n = N \times N \) Manhattan-street network, the total complexity of a node, \( C_{node} \), consists of the complexity of the shared buffers and its associated circuits, and the complexities of the IPP and OPP is estimated by:

\[
C_{node} \approx \left[ \kappa_s B d c_1 + 2 B d^2 c_2 \left( \frac{3N}{4} \right) + C_c \right] + [C_{ipp} + C_{opp}]. \tag{6.48}
\]

The first term represents a complexity of \( b \) shared buffer slots each storing \( \kappa_s = 53 \) bytes of the ATM cell with the transistor counts of \( c_1 \). We refer to \( B \) as the number of shared buffers per port of a node \((B = b/d)\) where the number of ports of each node is \( d = 3 \). The second term refers to the complexity of two \( d \times b \) crossbars each of which uses crosspoint switches with a complexity of \( c_2 \). \( C_c \) gives the cost of the node controller. In the second term, \( 3N/4 \) is the speed advantage. The speed advantage directly influences the number of transistors of the two crossbars of each node belonging to the shared buffers.

Note that, for the MSN, \( C_{net}/n = C_{node} \) since the network is regular and quadratic. Also note that the most complex parts of each node are the IPP and the OPP. With \( B = 3, n = N^2 = 16 \), and assuming \( c_1 = 12, c_2 = 24, C_c = 12K, \) and \( C_{ipp} + C_{opp} \approx 3.2M \) transistors, the total transistor cost per port is estimated to be 3.30M transistors.
Figure 6.9: Complexity of the network per port (in millions of transistors) vs the network size \( n \), and the number of shared buffers per port of node \( B \).

Figure 6.9 shows the complexity per port of network \( C_{\text{net}}/n \) where \( n \) is the total number of ports to the network (inputs or outputs).

### 6.7. Estimation of Area

In this section, the area of the MSN is evaluated where different technology feature sizes \( t \) are considered. Remember that, the increase of network size in the MSN would require an increase in the width of data paths and therefore a resulting increase of complexity. We use the results of the complexity evaluation presented in Section 6.6 to estimate the area required for the fabrication of the MSN on a single wafer.

According to the results of this section, each node consumes \( \kappa_2 B d c_1 + 2 B d^2 c_2 \left( \frac{3N}{4} \right) + C_c \) transistors plus 3.2M transistors for the IPP and the OPP. We assume each transistor occupies 400\( \lambda^2 \), where \( \lambda = t/2 \) is the minimum feature size and \( t \) is the technology feature size. By adding a small percentage of the total area transistors for external routing, we can estimate the area of an entire the MSN for the fabrication on a single wafer. Figure 5.17 shows the results of this study for a 64-port and a 256-port network. In this figure we also give the maximum area available on 5-, 6-, and 8-inch
Figure 6.10: Evaluation of the network area on a single wafer versus the technology feature size \((t)\).

wafers. For the estimation of the available area, we consider a large square at the center of a wafer and four narrow rectangles attached to each side of the square.

6.8. Evaluation of Yield for WSI Technology

In this section, the yield of the MSN nodes for using WSI technology is evaluated. The *negative binomial distribution* derived from Equation (4.4) is used to model the circuit integration yield of the MSN network. By substitution of the area required for each denoted by \(A_{\text{node}}\) in Equation (4.4), the overall yield for the MSN network is given by:

\[
y = \frac{1}{\left[1 + \frac{A_{\text{node}}D_0}{(\sigma/\mu)^2}\right](\sigma/\mu)^2}, \tag{6.49}
\]

The numerical results of the number of functional nodes in the network is estimated in Table 6.4. We are giving two types of results which include the yield for IPP plus OPP and the yield for the remaining parts of a node. The number of defects per square centimeter \(D_0\) is assumed to be 1, 2 and 3.
Table 6.4: Yield (y) evaluation for the node of the Manhattan street network.

<table>
<thead>
<tr>
<th></th>
<th>( B )</th>
<th>( D_0 = 1 )</th>
<th>( D_0 = 2 )</th>
<th>( D_0 = 3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>2</td>
<td>0.9797</td>
<td>0.9614</td>
<td>0.9413</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.9633</td>
<td>0.9480</td>
<td>0.9176</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.9506</td>
<td>0.9303</td>
<td>0.8864</td>
</tr>
<tr>
<td>OPP/OPP</td>
<td></td>
<td>0.8811</td>
<td>0.7675</td>
<td>0.6927</td>
</tr>
</tbody>
</table>

6.8.1. Defect Tolerance

In the previous sections we stated that the deflection strategy and the regularity in the MSN structure make it suitable for constructing a defect-tolerant system. There are two distinct major conditions under which the network would malfunction, defective nodes and defective links. If a node is found to be defective, it must be bypassed. In this way, the remainder of the loop continues to operate. Consider Figure 6.1 and assume that a cell being processed at a working node \( (i, j) \) needs to be transmitted to a defective node like \( (i, j + 1) \). Since node \( (i, j + 1) \) is bypassed, the cell is automatically routed to its successive node \( (i, j + 2) \). This leads to a longer-path penalty, but the cell will reach its destination. If a link between two nodes is open, the other links connected to this node can still operate, but the loop at which the node is located is broken. Assume one of the internal links of a node \( (i, j) \) is not usable. The penalty here is congested traffic at the node since all of the cells should be transmitted through the alternative link. Again, adopting these strategies against the defective components does not cause the entire system to malfunction, and instead a penalty of longer delay must be paid. This is one of the main advantages of using the MSN structure for the WSI technology.
6.9. Remarks

In the next chapter we will evaluate the performance of the network with simulations. The numerical results of the performance analysis of this chapter will be given together with the results of simulation studies and compared.
7. SIMULATION OF MANHATTAN STREET NETWORK

This chapter\(^1\) presents a simulation of the performance of the Manhattan-street network discussed in Chapter 6. We measure the system throughput, the average number of deflections and the average delay for various configurations. Extensive simulation experiments assess the effectiveness of the speed advantage required to accommodate arbitrary traffic on the external links. We evaluate the effects of high offered traffic loads, large shared queues, varying speed advantage, and increasing network size on the network performance.

This chapter is organized as follows; in Section 7.1 the structure of the simulator is discussed. In Section 7.2 the results of experiments on the system throughput, delay and average number of deflections are presented and compared with the results of the analysis.

7.1. Simulation Structure

The simulation is designed based on a stochastic model. The simulator is constructed from eight main routines: \texttt{initialize()}, \texttt{cellarrive()}, \texttt{msn.route()}, \texttt{contend()}, \texttt{deflect()}, \texttt{buffer()}, \texttt{flow.cell()} and \texttt{calc.stats()}. Different components such as arriving or leaving cells, nodes, ports, and buffer slots are modeled in this simulator. To capture the behavior of the network under various loads, we examine the effect of different variables. The program that computes the results consists of about 22 pages of C code. We briefly explain a few of the subprograms in this section.

\(^1\)This chapter together with Chapter 6 has been (or will be) published in part in [39], [40], [44].
7.1.1. Cell Arrival

In routine \texttt{cellarrive()} each input port of a node \((i,j)\) generates a new cell independently with a fixed probability of \(\rho\) as shown in Figure 7.1. The generation of cells is based on a Bernoulli arrival process. The arriving cells are independently assigned random destination addresses. A node also receives cells from its two predecessors \((i+1,j)\) and \((i,j-1)\). A total of \(b\) shared buffers is considered at a given node. Upon the generation of a cell, the length of the input queue is incrementally increased. However, if this length exceeds the maximum number of input buffers, cell loss occurs instead. At the next step, the presence of a cell becomes one but the cell still is not allowed to enter the network until permission is granted from the shared buffer to accept the cell. The destination of each cell is determined at random where in an \(n\)-port network \((N = \sqrt{n})\) both the row and column addresses belong to \(\{0,1,...,N-1\}\).
7.1.2. Routing Strategy

We explained in Section 6.1.1 of the previous chapter how cells flow in the MSN. The routing algorithm in the simulation tool is also based on the strategy discussed in that chapter. Recall that, for each cell, this algorithm partitions the network into four quadrants and four subareas with the destination of the cell realized approximately in the center of the network. With this mapping, cells approach the center of the network through preferred directions defined for each region.

In routine msn-route() of the simulator, the destination address of the cell is converted to (0,0) and its current address is converted to a relative address. The cell is then examined depending upon within what quadrant or subarea its current row and column addresses place it, and then its next row and column addresses are determined based on the preferred directions for the regions. The simulator also specifies the node's exit port number based on each region's preferred direction.

7.1.3. Contention and Deflection

Routine contend() simulates the contention of two or more cells to access an output. There are two distinct cases, contention for an external link, and contention for any of the internal outgoing links. If the buffer is full, the loser of the contention is deflected on the other link through a separate routine deflect(). Notice that deflection of cells occurs only on internal links. To implement the deflection, the next row and column addresses of a cell are compared with its current row and column addresses respectively, and new values for next row and column addresses and the node's exit port are determined.
7.1.4. Traffic Flow

In `flow.cell()`, cells are advanced in the network after testing two distinct values. First, a cell is examined to determine if a cell is designated to leave the node. Next, the location of a cell is checked to see if it has reached its destination. The latter is accomplished by a set of two tests to see if the current row and column addresses of the cell are equal to respectively its destination row and column addresses. Upon exit, the cell's timestamp is subtracted from the running clock and the result is added to the overall delay. At this point, the cell leaves the node and its buffer slot accepts a new cell.

A similar approach takes place at the input queue. In the input queue, cells are generated and timestamped using the current running clock. They are assigned random destination addresses and are allowed to enter the shared buffers when the flow control signals allow them to do so.

7.2. Simulation Results and Comparison with Analysis

This section describes some selected results from the extensive simulation experiments and compares them with the numerical results of the analysis presented in the previous chapter. The simulation was run under the uniform traffic assumption by which cells generated at the nodes were assigned random destination addresses with each destination address equally likely. We give the results of the average number of cells leaving the network per cell cycle per external link which is referred to as throughput ($T$). The average delay ($D$) is also measured from the time a cell enters the network to the time the cell leaves the network. To determine how much time a cell spends traveling the network, a cell is timestamped upon its arrival, and when the cell leaves the network the timestamp value is subtracted from the current time. By comparing the delay with the
length of the least-cost path from source to destination, we can assess how efficiently the network is operating. The average number of deflections (Δ) per exiting cell is another measurement by which various results of the simulation are verified. We evaluate the effects of increasing offered traffic loads, large shared queues, varying speed advantage, and increasing network size on the network performance. Speed advantage refers to the ratio of the internal link speed to the external link speed.

7.2.1. Collected Data on the Number of Flowing Cells

It was mentioned in various previous discussions that any flowing cell in the network was identified by a “type” depending on the cell’s next output address. We classified cells into types 1, 2, 3, and 4 referring to the cells being respectively addressed to external links, row internal links, column internal links, and either row or column internal links. With the simulation, we collected corresponding data on the number of cells of type 2, 3 and 4 listed in Tables 7.1 through 7.5. These types of cells for which some statistics have been predicted through the analytic approach of the performance evaluation in Section 6.3 are of particular interest.

Table 7.1 provides the numerical results obtained from the simulation of a 16-node Manhattan street network with 2 shared buffer slots per port of each node (B = 2). The cells were counted through several simulation runs with different rates of external offered load, ρ. The data shown in Table 7.1 was collected within quadrant Q_1 of the network mapped in Figure 6.2. The reported statistics indicate that the number of cells of type 2 is almost the same as the the number cells of type 3 and each is approximately 25% of the total number of cells of types 2, 3, and 4, while the number of cells of type 4 includes 50% of the total. The percentage of type-4 cells was expected since exactly in 50% of nodes in quadrant Q_1, cells can be addressed to either internal outputs. We
Table 7.1: Collected data from the simulation of a 16-node network: the number of cells of type 2, 3, and 4 from quadrant $Q_1$.

<table>
<thead>
<tr>
<th>offered load ($\rho$)</th>
<th>type-2</th>
<th>type-3</th>
<th>type-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1228</td>
<td>1182</td>
<td>2373</td>
</tr>
<tr>
<td>0.2</td>
<td>1789</td>
<td>1923</td>
<td>3652</td>
</tr>
<tr>
<td>0.3</td>
<td>2656</td>
<td>2743</td>
<td>5445</td>
</tr>
<tr>
<td>0.4</td>
<td>3642</td>
<td>3743</td>
<td>7585</td>
</tr>
<tr>
<td>0.5</td>
<td>4652</td>
<td>4577</td>
<td>9420</td>
</tr>
<tr>
<td>0.6</td>
<td>5472</td>
<td>5439</td>
<td>11387</td>
</tr>
<tr>
<td>0.7</td>
<td>5573</td>
<td>5615</td>
<td>11498</td>
</tr>
<tr>
<td>0.8</td>
<td>5565</td>
<td>5560</td>
<td>11366</td>
</tr>
<tr>
<td>0.9</td>
<td>5559</td>
<td>5520</td>
<td>11143</td>
</tr>
<tr>
<td>1.0</td>
<td>5548</td>
<td>5514</td>
<td>11019</td>
</tr>
</tbody>
</table>

| percentage, Simulation | 25.12% | 24.97% | 49.90% |
| percentage, Analysis   | 25.00% | 25.00% | 50.00% |

Table 7.2: Collected data from the simulation of a 16-node network: the number of cells of type 2, 3, and 4 from quadrant $Q_2$.

<table>
<thead>
<tr>
<th>offered load ($\rho$)</th>
<th>type-2</th>
<th>type-3</th>
<th>type-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1213</td>
<td>2301</td>
<td>1268</td>
</tr>
<tr>
<td>0.2</td>
<td>1820</td>
<td>3623</td>
<td>1884</td>
</tr>
<tr>
<td>0.3</td>
<td>2618</td>
<td>5429</td>
<td>2735</td>
</tr>
<tr>
<td>0.4</td>
<td>3617</td>
<td>7533</td>
<td>3525</td>
</tr>
<tr>
<td>0.5</td>
<td>4639</td>
<td>9531</td>
<td>4520</td>
</tr>
<tr>
<td>0.6</td>
<td>5521</td>
<td>11399</td>
<td>5432</td>
</tr>
<tr>
<td>0.7</td>
<td>5532</td>
<td>11478</td>
<td>5498</td>
</tr>
<tr>
<td>0.8</td>
<td>5455</td>
<td>11378</td>
<td>5401</td>
</tr>
<tr>
<td>0.9</td>
<td>5429</td>
<td>11256</td>
<td>5382</td>
</tr>
<tr>
<td>1.0</td>
<td>5411</td>
<td>11190</td>
<td>5362</td>
</tr>
</tbody>
</table>

| percentage, Simulation | 25.12% | 49.90% | 24.97% |
| percentage, Analysis   | 25.00% | 50.00% | 25.00% |
Table 7.3: Collected data from the simulation of a 16-node network: the number of cells of type 2, 3, and 4 from quadrant $Q_3$.

<table>
<thead>
<tr>
<th>offered load ($\rho$)</th>
<th>type-2</th>
<th>type-3</th>
<th>type-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1198</td>
<td>1182</td>
<td>1173</td>
</tr>
<tr>
<td>0.2</td>
<td>1877</td>
<td>1901</td>
<td>1852</td>
</tr>
<tr>
<td>0.3</td>
<td>2638</td>
<td>2623</td>
<td>2645</td>
</tr>
<tr>
<td>0.4</td>
<td>3572</td>
<td>3623</td>
<td>3618</td>
</tr>
<tr>
<td>0.5</td>
<td>4652</td>
<td>4577</td>
<td>4420</td>
</tr>
<tr>
<td>0.6</td>
<td>5463</td>
<td>5443</td>
<td>5487</td>
</tr>
<tr>
<td>0.7</td>
<td>5570</td>
<td>5590</td>
<td>5599</td>
</tr>
<tr>
<td>0.8</td>
<td>5545</td>
<td>5562</td>
<td>5550</td>
</tr>
<tr>
<td>0.9</td>
<td>5529</td>
<td>5540</td>
<td>5532</td>
</tr>
<tr>
<td>1.0</td>
<td>5501</td>
<td>5521</td>
<td>5516</td>
</tr>
</tbody>
</table>

| percentage, Simulation | 33.36% | 33.36% | 33.35% |
| percentage, Analysis   | 33.33% | 33.33% | 33.33% |

Table 7.4: Collected data from the simulation of a 16-node network: the number of cells of type 2, 3, and 4 from quadrant $Q_4$.

<table>
<thead>
<tr>
<th>offered load ($\rho$)</th>
<th>type-2</th>
<th>type-3</th>
<th>type-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2378</td>
<td>1193</td>
<td>1202</td>
</tr>
<tr>
<td>0.2</td>
<td>3653</td>
<td>1892</td>
<td>1922</td>
</tr>
<tr>
<td>0.3</td>
<td>5423</td>
<td>2772</td>
<td>2795</td>
</tr>
<tr>
<td>0.4</td>
<td>7542</td>
<td>3783</td>
<td>3745</td>
</tr>
<tr>
<td>0.5</td>
<td>9356</td>
<td>4507</td>
<td>4499</td>
</tr>
<tr>
<td>0.6</td>
<td>11367</td>
<td>5389</td>
<td>5385</td>
</tr>
<tr>
<td>0.7</td>
<td>11473</td>
<td>5575</td>
<td>5565</td>
</tr>
<tr>
<td>0.8</td>
<td>11378</td>
<td>5568</td>
<td>5526</td>
</tr>
<tr>
<td>0.9</td>
<td>11201</td>
<td>5529</td>
<td>5483</td>
</tr>
<tr>
<td>1.0</td>
<td>11190</td>
<td>5502</td>
<td>5459</td>
</tr>
</tbody>
</table>

| percentage, Simulation | 50.51% | 24.83% | 24.64% |
| percentage, Analysis   | 50.00% | 25.00% | 25.00% |
Table 7.5: Collected data from the simulation of a 64-node network: the number of cells of type 2, 3, and 4 from quadrant $Q_1$.

<table>
<thead>
<tr>
<th>offered load ($\rho$)</th>
<th>type-2</th>
<th>type-3</th>
<th>type-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>4138</td>
<td>4257</td>
<td>8300</td>
</tr>
<tr>
<td>0.2</td>
<td>8161</td>
<td>8369</td>
<td>16407</td>
</tr>
<tr>
<td>0.3</td>
<td>11430</td>
<td>11511</td>
<td>23866</td>
</tr>
<tr>
<td>0.4</td>
<td>14537</td>
<td>14441</td>
<td>29954</td>
</tr>
<tr>
<td>0.5</td>
<td>17330</td>
<td>17042</td>
<td>35704</td>
</tr>
<tr>
<td>0.6</td>
<td>20083</td>
<td>21599</td>
<td>43950</td>
</tr>
<tr>
<td>0.7</td>
<td>24978</td>
<td>24681</td>
<td>48409</td>
</tr>
<tr>
<td>0.8</td>
<td>25962</td>
<td>25619</td>
<td>50683</td>
</tr>
<tr>
<td>0.9</td>
<td>25568</td>
<td>25446</td>
<td>50393</td>
</tr>
<tr>
<td>1.0</td>
<td>25409</td>
<td>25352</td>
<td>50201</td>
</tr>
<tr>
<td>percentage, Simulation</td>
<td>25.16%</td>
<td>25.11%</td>
<td>49.72%</td>
</tr>
<tr>
<td>percentage, Analysis</td>
<td>25.00%</td>
<td>25.00%</td>
<td>50.00%</td>
</tr>
</tbody>
</table>

Table 7.6: Comparison of simulation and analysis on the overall percentage of "type-4" cells (in the analysis we use the equivalent term of "don't-care nodes").

<table>
<thead>
<tr>
<th>network size</th>
<th>simulation</th>
<th>analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 16$</td>
<td>32.82%</td>
<td>33.33%</td>
</tr>
<tr>
<td>$n = 64$</td>
<td>38.98%</td>
<td>39.68%</td>
</tr>
</tbody>
</table>

notice here that a similar result can be obtained from the analysis presented in Table 6.1 of Section 6.3, by dividing $\nu_{Q_1}$ by the number of nodes in each quadrant ($N^2/4$) which is 2/4.

We also collected data for three other quadrants and compared the average percentage of cells of various types with the analysis shown in Tables 7.2, 7.3, and 7.4. Since the destination is always in quadrant $Q_3$, we note that the total number of nodes in this region is $(N/2)^2 - 1$ which is 3/4 the number for the other quadrants when $n = 16$.

Table 7.5 gives the results of a similar simulation run but for a 64-node network. We finally, in Table 7.6, compare the overall percentage of "type-4" cells obtained from
the simulation with the overall percentage of the "don't-care nodes" presented in the analysis results of Table 6.1. The table reports the data for 16-node and 64-node networks based on the entire network.

7.2.2. Effect of Increasing Offered Traffic Load

The next set of results includes Figures 7.2, 7.3, and Table 7.7. These results compare the numerical outputs of the analysis given in the previous chapter with the simulation. Note that these results do not include the effect of speed advantage at this point. Figure 7.2 shows the throughput with three different numbers of buffer slots per port of each node \( (B) \). With \( B = 2 \) and \( n = 16 \), at offered loads exceeding \( \rho = 0.4 \), the curve starts to flatten out since the network begins to reach its full capacity.

By further increasing the load, the throughput does not improve further and even falls slightly since the growing incoming traffic increases the number of deflections as shown in Table 7.7. The flattening curves in Figure 7.3 give the delay through the network, while the rising curves add the delay in the input queue. Notice the delay becomes large when the network capacity is reached. It is also worth noting that most of the total delay is incurred within the input queue when the load is heavy. Finally, we compare the average number of deflections \( (\Delta) \) obtained from the simulation and analysis. We notice a loop correlation effect at large offered loads in Figure 7.2 and Table 7.7 where the decrease of throughput slows down the increase of deflections, thus avoiding further decline of the throughput.

Higher throughputs can be obtained using larger buffers. Both the simulation and analysis show an improvement in the throughput when the size of the shared buffer increases. However, the overall throughput cannot exceed an upper bound as seen in Figure 7.2. This mainly expresses the maximum capability of the network in handling
Figure 7.2: Throughput ($T$) versus offered load ($\rho$) when the number of shared buffers per port of node ($B$) varies.

Figure 7.3: Delay ($D$) vs offered load ($\rho$) when the number of shared buffers per port of node ($B$) varies.
Table 7.7: Comparison of simulation and analysis on the average number of deflections ($\Delta$) versus offered load ($\rho$) when the number of shared buffers per port of node ($B$) vary.

| offered load ($\rho$) | analysis | | simulation |
|---|---|---|---|---|---|---|
| | $B = 2$ | $B = 3$ | $B = 4$ | $B = 2$ | $B = 3$ | $B = 4$ |
| 0.1 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| 0.2 | 0.00287 | 0.00000 | 0.00000 | 0.00324 | 0.00000 | 0.00000 |
| 0.3 | 0.01492 | 0.00094 | 0.00000 | 0.01689 | 0.00119 | 0.00000 |
| 0.4 | 0.06827 | 0.01342 | 0.00289 | 0.07583 | 0.01463 | 0.00315 |
| 0.5 | 0.16270 | 0.05229 | 0.03031 | 0.18739 | 0.05840 | 0.03384 |
| 0.6 | 0.27219 | 0.15430 | 0.10842 | 0.31072 | 0.17829 | 0.12156 |
| 0.7 | 0.40320 | 0.28902 | 0.22082 | 0.45424 | 0.31027 | 0.24673 |
| 0.8 | 0.56009 | 0.42964 | 0.36183 | 0.62049 | 0.46047 | 0.39090 |
| 0.9 | 0.63034 | 0.49648 | 0.41461 | 0.70302 | 0.56844 | 0.46931 |
| 1.0 | 0.65884 | 0.51038 | 0.42996 | 0.72171 | 0.57942 | 0.47338 |

the traffic at this rate of offered load. As was expected, by enlarging the capacity of the shared queue, the number of deflections is reduced as well. We will show later that a speed advantage is needed to improve performance further. Comparisons on the results of simulation and the analysis exhibit admissible accuracy in the performance evaluation as seen in the plots.

7.2.3. Effect of Network Size

The next set of plots given in Figures 7.4 and 7.5 focuses on the effect of the network size on the throughput. In Figure 7.4, we have examined the impact of different network dimensions including $n \in \{16, 64, 144, 256\}$ ($N \in \{4, 8, 12, 16\}$). In these experiments, the offered load is fixed at $\rho = 0.9$. As expected, enlarging the network size $n$ reduces the throughput. In this case, the system performance is sensitive to adding more buffer slots to the shared queue, but improving the system throughput with this approach is impractical and it works out only to a certain point as explained earlier.
Figure 7.4: The effect of enlarging network size ($n$) on throughput ($T$) for three different numbers of buffers per port of node ($B$) when the speed advantage is one.

Figure 7.5: The effect of enlarging network size ($n$) on throughput ($T$) for three different numbers of buffers per port of node ($B$) when the speed advantage is $3N/4$ ($n = N^2$).
Figure 7.5 clearly demonstrates the effect of network size on the throughput when the speed advantage is $3N/4$. We used this value since the average number of hops in the networks we simulated is close to this figure. Note that the speed advantage varies with the network size, thus enlarging the network dimension has little impact on the throughput as shown in the figure.

7.3. Other Results of Simulation

In the next two sections we present some other selected results of the simulation.

7.3.1. Effect of Large Shared Buffers

We evaluate the effect of higher numbers of shared buffers per port of node ($B$) on the network performance shown in Figures 7.6, 7.7, and 7.8. The curve labeled by $\rho = 0.1$ in Figure 7.6 clearly shows that the network is fully capable of handling traffic at low loading level and thus larger numbers of internal buffers have no impact on performance in this case. Notice that at $\rho = 0.1$ no deflections occur since the offered load is too low.

At the midrange offered load of $\rho = 0.5$, increasing $B$ improves the throughput. As Figure 7.8 indicates, with increasing $B$, the average number of deflections is reduced. It is seen that decreasing deflections continues up to $B = 7$. Beyond $B = 7$, the network queueing capacity appears to be large enough but buffers eventually become full. In this case, according to Figure 7.6, the system transmits the offered load with its nearly full capability to the external links, the number of deflections stays constant at its low value, and delay continues to rise.

At the heavier traffic load of $\rho = 0.9$, we observe a similar behavior. As expected, increasing the number of shared buffers raises the throughput to a certain point, and
Figure 7.6: The effect of increasing the number of shared buffers per port of node \( B \) on throughput \( T \) when offered load \( \rho \) varies.

Figure 7.7: The effect of increasing the number of shared buffers per port of node \( B \) on delay \( D \) when offered load \( \rho \) varies.
Figure 7.8: The effect of increasing the number of shared buffers per port of node ($B$) up to 10 on the average number of deflections ($\Delta$) for three different offered loads ($\rho$).

consequently increases the delay. Obviously, for the same reason as explained in the previous case, the average number of deflections goes down which is seen in Figure 7.8. Since the traffic load is high in this case, at around $B = 7$ the network approaches its maximum capability in transmitting traffic. It is shown in Figure 7.6 that the throughput cannot exceed this limit. At this point, increasing the buffer size further does not improve the throughput. In other words, having larger buffers at this offered load leads to the accumulation of a larger number of cells in the queue which simply causes more delay as is shown in Figure 7.7. We recall that with buffer sizes $B = 2$ and $B = 3$, the throughput for $\rho = 0.9$ is lower than the throughput for a load in the range of $0.6 \leq \rho < 0.9$. It was explained before that this behavior is due to the increase of deflections at high loads. The throughput with the larger buffer sizes, however, gets further improved compared to the case with $\rho = 0.5$. 
Figure 7.9: The effect of speed advantage ($s$) on the throughput ($T$) of the network when offered load ($\rho$) varies ($n = 16$).

7.3.2. **Effect of the Speed Advantage**

In the previous discussion, we observed that the network seems to be incapable of handling high traffic loads. We also noticed that increasing the number of shared buffers can improve the throughput, but this approach is useful only to a certain point beyond which the throughput even decreases.

The next set of results shown in Figures 7.9 and 7.10 evaluates the effectiveness of a speed advantage ($s$) on the network performance. The simulation repeats all previous experiments but letting the internal data paths operate at a rate $s$ times faster than the rate of the external links where $s \in \{2, 3, ..., 10\}$. The results are given together with the previous observation of rate $s = 1$. With the speed advantage of 2 and 3, throughput
Figure 7.10: The effect of speed advantage ($s$) on the throughput ($T$) of the network when offered load ($\rho$) varies ($n = 16$).

improves approaching the ideal case. The plots show that in the neighborhood of $s = 5$, that is $s = 1.25N$, the network performs ideally meaning that it delivers the entire offered load to external links regardless of the traffic load.
8. CONCLUSIONS

In this dissertation we have proposed and analyzed different switching architectures aimed at high-speed ATM applications. This research focused on the speed, performance, and scalability of the switching systems, and an effort was made to improve the systems from these perspectives to support the needs of future communication networks. The work at the network level concentrated on both the technology type and the architecture type in order to achieve high bandwidths. Contributions of this dissertation were as follows.

8.1. Summary and Contributions

Chapter 2 presented an overview of broadband ATM switching systems. The basic operation of a typical ATM system was briefly explained. We also presented a survey of ATM switching systems and classified the switching network topology used in ATM networks.

In Chapter 3, an ATM multipoint switch supporting gigabit per second links was presented where the system design was based on VLSI technology. We presented architectural aspects, switch level design, switch level timing simulation, multicast and routing algorithms, traffic performance analysis, and evaluation of complexity. The system uses two Beneš networks for routing and copying cells. Each switch element consists of parallel data-slices with shared buffering to achieve the desired bandwidth. A local controller determines which buffered cells are to be sent to the outputs and also generates grant flow controls through the network to avoid the overflow of buffers.
The circuit simulations were set up for the internal VLSI circuitry of the switch controller in order to understand the system's timing performance at the clock speed of 100 MHz. The timing simulation for the access of two competing ATM cells for a desired output was studied, and the design issues at a relatively high speed clock were discussed. The simulation results of the switch controller at the clock speed of 100 MHz were presented. One way to increase the system capability for higher speeds is to increase the number of parallel planes to let the internal circuitry of the fabric handle a larger volume of traffic in a given time slot.

Given the performance of the architecture presented in Chapter 3, Chapter 4 extended the discussion on the enhancement of the speed for switching systems through the use of WSI technology. WSI is especially attractive for switching networks since the length of interconnections is minimized. We evaluated the possibility of using wafer-scale integration technology for typical ATM switching systems. The evaluation of the multistage switches designed for the VLSI technology indicated that a considerable portion of a semiconductor wafer must be consumed for the implementation of interconnections if they are considered for the WSI technology. Although, a multistage switch can be an option for using WSI, we studied alternative architectures that would make better sense for WSI technology when the regularity in structure and simplicity of testing are important.

In Chapter 5, we presented a novel switching system architecture which takes advantage of the technological characteristics of wafer-scale integration. The network consists of a multipath buffered crossbar, and employs a simple cell routing strategy which is suitable for handling traffic at a high speed. The structure of this switch avoids conventional interconnection structures to increase speed and traffic throughput. In addition to these advantages, the inherent network structure can effectively simplify the
scheme for defect tolerance and therefore, system reconfiguration. For the architecture proposed in this chapter, we developed an analytical queueing model and used it to evaluate the performance of the system. The results demonstrated significant potential for high performance. We also evaluated the complexity, yield, and area of the system. The evaluation of area indicated that networks at the scale of 64-ports up to 256-ports can be entirely implemented on a single wafer depending on the technology and the wafer size.

In Chapters 6 and 7, we proposed the adaptation of the Manhattan-street network (MSN) for use as an ATM switch fabric, because of its natural suitability for wafer-scale integration. We proposed using a shared buffer to achieve the best possible queueing performance in the least area. We developed analytical and simulation models for the MSN and used them to study the traffic performance of this network. The key new result emerging from this study was the need to scale the bandwidth of the MSN links in proportion to the square root of the number of switch ports. From a technological perspective, the network architecture has a special capability for fault tolerance since a faulty node or link does not cause the entire system to be disabled. This feature makes it suitable for constructing a defect tolerant wafer-scale ATM switching system. We evaluated the complexity per port of the wafer-scale system and showed the yield for parts of each node.

8.2. Comparison of Wafer-Scale Switches

We can now compare the networks presented in this dissertation from different standpoints. Improvement in the structure of switching systems for higher speed applications was achieved based on the practical issues understood from the multistage switch presented in Chapter 3. The internal switch structure and especially the buffer controller mechanism presented in this chapter were adopted for use in the other systems. The
multistage switch and the MSN both used shared buffering with a slightly different buffer controller. The crosspoint of the MBC network was constructed with a similar buffer control circuit and contention bus structure.

At the network level, the performance evaluations indicated that the delay in the MBC network was the lowest compared to the MSN and the multistage network. This is obviously due to the crossbar structure of the MBC and the fact that a cell only passes through two buffering nodes. The delay curves in both the MBC and multistage networks flattened out eventually after the offered traffic load exceeded the queueing capacity. The throughput of point-to-point connections in the MBC network was considerably higher than the two others. The reason is the two-node-routing-connection structure of the MBC.

From the complexity point of view, between the two wafer-scale systems, the MSN consumed fewer components than the MBC. The other advantage of the MSN was that it had a fully regular structure with which a defect-tolerant network would be constructed more simply. The research in this dissertation showed that the fabrication of large-scale systems like a 256-port MSN network on a single 6-inch wafer may be feasible with currently available technologies while fabrication of a 256-port MBC network requires larger wafers. We estimated the yield of these two systems as well. The study indicated that the percentage of good nodes not including the port processors in the MSN is lower than the percentage of functioning crosspoints in the MBC. This result would have been predicted since the MSN typically requires larger buffers.

8.3. Extension and Future Work

There are several questions related to this research that remain to be explored.
The first extension to this research should be to analyze the Manhattan street network with bidirectional links. This architecture uses a similar node structure as proposed for the unidirectional Manhattan street network and could potentially be a higher-performance system since the routing algorithm is much simpler. In addition to this advantage, bidirectional links could provide higher bandwidths. This behavior is due to the higher accessibility of cells to nodes. On the other hand, bidirectional links clearly increase the circuit complexity significantly, and it is not entirely clear whether it would yield an improvement overall.

Another extension associated with the work presented for the Manhattan street network is to develop practical ways to achieve the speed advantage that the analysis shows is necessary. One way to efficiently provide the speed advantage is to use wide data paths at each MSN node. In addition to this method, there are other architectural possibilities that can be realized such as increasing the number of nodes with respect to the number of port processors. It would also be interesting to study the MSN performance when some random fraction of nodes have failed, so as to determine what additional speed advantage is needed in this case to ensure adequate performance.

For the performance evaluation work, examining the other traffic types like bursty traffic is desirable. This extension is particularly important in order to understand the reliability of the switches when they face more realistic traffic.

For the wafer-scale systems, constructing algorithms for testing entire systems is an open subject. This dissertation focused on the other parameters like architecture, speed, performance, complexity, yield, simulations, scalability, and area since at this level of research these factors seemed to be more important. Together with the testing, estimation of cost may be worth considering. While our understanding is that the cost of wafer-scale switches can be much lower than those manufactured in a traditional
approach using the VLSI technology and printed circuit boards, we avoided discussions on cost throughout this thesis. So far, no individual or company has publicized a clear estimate on a commercialized wafer-scale system. Thus, we preferred not to comment on the cost of wafer-scale switching networks that were proposed in this dissertation. However, there is strong optimism in the literature that WSI technology is a potential approach to considerably reducing the cost, and in fact this was the motivation for using such a technology in the first place.

The other engineering issues in a wafer-scale system to be investigated include: clock distribution, the upper bound of the length of lines on the wafer, power consumption and cooling in WSI, off-wafer interconnection design, organizational arrangements on the wafer, and system reconfiguration methods. For understanding how a wafer-scale system should be reconfigured, one would need to study the modern method of RVLSI1 [2].

Also, a new subject of multiwafer module technology, where large systems with multiple interconnected wafers can be investigated for networking. In particular, for the application of such an idea, the torus structure of the Manhattan street network is favorable to building a very large scale and easily scalable network. In this case, interconnected multiple wafers are stacked together and cooling layers are sandwiched in between.

1laser restructurable VLSI
ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. Jonathan S. Turner, for all of the invaluable help and guidance he has given me in the development of this thesis. Over the past five years, I have continuously received insight through his intellectual dialogue and invaluable help through his extensive scientific/engineering knowledge. I feel truly privileged to have worked with him. This thesis is dedicated to him.

I am thankful to Professors Roger Chamberlain, Robert Morley, David Richard and Fred Rosenberger for serving on my dissertation committee and providing useful comments. I would also like to thank Professor Barry Spielman, the chairman of the electrical engineering department, for his encouragement and support throughout my graduate studies at Washington University.

I wish to acknowledge the sponsors of this research, the National Science Foundation, Ascom Timplex, Bell Communications Research, Bell Northern Research, Digital Equipment Corporation, Italtel SIT, NEC America, NTT, SynOptics Communications, and Tektronix.

Thanks to everyone in the Computer and Communications Research Center (CCRC) at Washington University, Dr. Ken Wong who has always been a great help, and all my friends: Andy, Christos, Chuck, Ehab, Einir, Girish (a \LaTeX expert), Gopal, Mahesh, Apostolos, Rex, Shree, Tienyo, Vasudha, Xiaolin, Zubin, and especially Hari, Greg Peterson and my office-mate Brad Noble.

I would like to extend my appreciation to the staff of the electrical engineering and computer science departments: Jean, Lisa, Mary, Myrna, Rita, Sharon, and Susan who have done much to help me over the past six years.
A special thanks goes out to Mr. Parviz Karandish, who has been a great support for me in St. Louis. When I needed time off from my research, he graciously accepted to accompany me with violin as I played piano.

Finally, I offer my most heartfelt thanks to my parents who have taught and guided me with optimism throughout my life. Their unconditional faith and endless love in me were indispensable to the completion of this thesis.
10. **ACRONYMS AND SYMBOLS**

10.1. Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>arriving-cell counter</td>
</tr>
<tr>
<td>AE</td>
<td>arbitration element</td>
</tr>
<tr>
<td>ATM</td>
<td>asynchronous transfer mode</td>
</tr>
<tr>
<td>BISDN</td>
<td>broadband integrated services digital network</td>
</tr>
<tr>
<td>BBC</td>
<td>buffer-control bit circuit</td>
</tr>
<tr>
<td>BCC</td>
<td>buffer control circuit</td>
</tr>
<tr>
<td>BCN</td>
<td>broadcast channel numbers</td>
</tr>
<tr>
<td>BS</td>
<td>bit slice</td>
</tr>
<tr>
<td>BTC</td>
<td>broadcast translation circuit</td>
</tr>
<tr>
<td>CCITT</td>
<td>international consultative committee for telecom and telegraphy</td>
</tr>
<tr>
<td>CD</td>
<td>column decoder</td>
</tr>
<tr>
<td>CMOS</td>
<td>complimentary metal-oxide semiconductor</td>
</tr>
<tr>
<td>CN</td>
<td>copy network</td>
</tr>
<tr>
<td>CRC</td>
<td>contention resolution circuit</td>
</tr>
<tr>
<td>DS</td>
<td>data slice</td>
</tr>
<tr>
<td>FOL</td>
<td>fiber optic link</td>
</tr>
<tr>
<td>GSS</td>
<td>grant signal selector</td>
</tr>
<tr>
<td>HD</td>
<td>header decoder</td>
</tr>
<tr>
<td>IPP</td>
<td>input port processor</td>
</tr>
<tr>
<td>LBC</td>
<td>local buffer controller</td>
</tr>
<tr>
<td>MBC</td>
<td>multipath buffered crossbar</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>MSN:</td>
<td>Manhattan street network</td>
</tr>
<tr>
<td>NDS:</td>
<td>network data slices</td>
</tr>
<tr>
<td>OCC:</td>
<td>output control circuit</td>
</tr>
<tr>
<td>OD:</td>
<td>output port decoder</td>
</tr>
<tr>
<td>OPP:</td>
<td>output port processor</td>
</tr>
<tr>
<td>PFC:</td>
<td>priority field circuit</td>
</tr>
<tr>
<td>RCB:</td>
<td>receive buffer</td>
</tr>
<tr>
<td>RCYC:</td>
<td>recycling buffer</td>
</tr>
<tr>
<td>RN:</td>
<td>routing network</td>
</tr>
<tr>
<td>RNG:</td>
<td>row number generator</td>
</tr>
<tr>
<td>RPC:</td>
<td>request priority circuit</td>
</tr>
<tr>
<td>RSQ:</td>
<td>resequencing buffer</td>
</tr>
<tr>
<td>RVC:</td>
<td>request vector circuit</td>
</tr>
<tr>
<td>SA:</td>
<td>synchronous adder</td>
</tr>
<tr>
<td>SBC:</td>
<td>shared buffer crossbar</td>
</tr>
<tr>
<td>SDC:</td>
<td>self-driven crosspoint</td>
</tr>
<tr>
<td>UGC:</td>
<td>upstream grant circuit</td>
</tr>
<tr>
<td>VLSI:</td>
<td>very large scale integration</td>
</tr>
<tr>
<td>VCI:</td>
<td>virtual circuit identifier</td>
</tr>
<tr>
<td>VXT:</td>
<td>virtual circuit translation table</td>
</tr>
<tr>
<td>WSI:</td>
<td>wafer-scale integration</td>
</tr>
<tr>
<td>WTG:</td>
<td>waiting time generator</td>
</tr>
<tr>
<td>XMB:</td>
<td>transmit buffer</td>
</tr>
</tbody>
</table>
Symbols

The following symbols are commonly used among the networks.

\begin{align*}
    d & : \text{number of inputs/outputs of a switch element} \\
    n & : \text{number of inputs/outputs of the network} \\
    \alpha & : \text{number of buffer slots in an input buffer} \\
    b & : \text{total number of buffer slots in a switch element (or queue)} \\
    \gamma & : \text{number of buffer slots in an output buffer} \\
    B & : \text{number of buffer slots per port of a switch element} \\
    \rho & : \text{offered traffic load to an input buffer} \\
    T & : \text{network throughput} \\
    D & : \text{network delay} \\
    Q_x & : \text{queue length of a switch element} \\
    C_{\text{net}} & : \text{complexity of the network per port of switch} \\
    y & : \text{yield}
\end{align*}
BIBLIOGRAPHY


VITA

• Born in Shiraz, Iran.

• Received B.Sc. in Electrical Engineering from Tehran University of Technology, Tehran, Iran, August 1985.
  – Received scholarship to pursue B.Sc. degree from Tehran University of Technology, Tehran, Iran.

• Received M.Sc. in Electrical Engineering from Washington University, St. Louis, MO, U.S.A, December 1990.
  – Received research assistantship to pursue M.Sc. degree from McGill University, Montreal, PQ, Canada.
  – Received research assistantship to pursue M.Sc. and D.Sc. degrees from Washington University, St. Louis, MO, U.S.A.

• Received D.Sc. in Electrical Engineering from Washington University, St. Louis, MO, U.S.A, May 1995.

• Publications: see [35] through [44].

• Member of the IEEE.