PERFORMANCE ANALYSIS, DESIGN AND RELIABILITY OF THE BALANCED GAMMA NETWORK

YASER EL SAYED

Performance Analysis, Design and Reliability of the Balanced Gamma Network

by

©Yaser El Sayed, B.Sc., M.Sc.

A thesis submitted to the School of Graduate
Studies in conformity with the requirements for the
Degree of Doctor of Philosophy

Faculty of Engineering and Applied Science
Memorial University of Newfoundland
December 1999

St. John's Newfoundland Canada

Abstract

Switching is one of the bottlenecks restraining the efforts of researchers toward implementing broadband communication systems. In this dissertation, we provide a comprehensive study of a promising switching architecture called the Balanced Gamma (BG) network. The BG network has shown good performance in terms of throughput, average cell delay, and reliability, and has displayed potential for application in broadband communications switch fabrics.

Designing highly reliable systems is a crucial requirement in the industry of broadband communications where consequences of the system failures are very expensive. Accordingly, we provide an exact model for network reliability of the BG network. The model demonstrates that the network is highly reliable and can be confidently deployed in communication systems.

The performance of the network is further investigated under different payloads containing uniform and non-uniform traffic. Uniform random and bursty are the traffic types used. Several simulation experiments are carried out to measure the cell loss, cell average delay, and buffering requirements of the BG network. In addition, we pursue an analytical model under uniform random traffic to verify our simulation results. The performance of the network is compared with both an ideal nonblocking network and the crossbar network. It is determined that the network has much better behavior than the crossbar switch and operates very closely to the ideal architecture under most types of offered traffic loads.

Finally, we introduce a VLSI design for the BG network using 0.35 CMOS technology supported by the Canadian Microelectronics Corporation. The design has mainly three components, the switching element, the output port, and the network main controller. The design features built-in self-test (BIST) which has become an essential part of any fast digital system. We also parametrize the design such that the amount of effort needed to generate a fabric with arbitrary size is minimal. We describe the design in the Very High Speed Integrated Circuit Description Language (VHDL).

Acknowledgments

First of all, I am thankful to The Almighty God, Who in His infinite mercy have helped me to bring this work to light.

I also owe a lot to my family, especially my mother and father for their continuous support and great sacrifices that were the major factors in making this work a reality.

I like to express my sincere thanks to my supervisor Dr. Venkatesan for his supervision, support, and cooperation in the course of this work. I also do not forget the support and help of my supervisory committee Dr. Howard Heys and Dr. Paul Gillard.

I like to thank the School of Graduate Studies at the Memorial University of Newfoundland for the financial support it provided during my Ph.D. program.

I will always remember the important discussions I had with Dr. Rod Byrne in the Department of Computer Science. These discussions have greatly enabled me in the design phase of the this work. Also, Mr. Michael Rendell in the Department of Computer Science deserves special thanks for his true cooperation in fixing the problems related to the VLSI CAD tools.

Finally, I like to thank all my friends and colleagues that sincerely helped during my Ph.D. program. I specially thank B. Balasubramanian, A. Khan, P. Mehrotra, K. Momin, F. Power, R. Thuppal, V. Vujjeni, and A. M. Zeiner.

Contents

List of Figures

List of Tables

Notation and List of Abbreviations

n	: number of stages in a MIN
B_{out}	: output buffer size
BR	: broadcast reliability
D_{in}	: delay in pure-input buffering strategy
D_{out}	: delay in pure-output buffering strategy
IL_i	: intput link i of an SE
K	: number of data planes is a switching network
L	: average burst length
N	: number of inputs/outputs in a rectangular switching network
NR	: network reliability
OL_i	: output link i of an SE
P	: number of service priorities
TP	: Throughput
TR	: terminal reliability
α	: switching parameter for the old routing decision of the BG network
λ	: failure rate
AHQ	: Address Holding Queues
ATE	: Automatic Test Equipment
ATM	: Asynchronous Transfer Mode
BB	: Batcher Banyan
BG	: Balanced Gamma
B-ISDN	: Broadband Integrated Services Digital Network
BIST	: Built-In Self-Test
BP	: backpressure
CAC	: Connection Admission Control
CMC	: Canadian Microelectronics Corporation
CMOS	: Complementary Metal Oxide Semiconductor
CP	: Control Port
CUT	: Circuit Under Test
DPO	: Delayed Pushout
DSM	: Deep Submicron
ECL	: Emitter Coupled Logic
EFCI	: Explicit Forward Congestion Indication
EXCON	: Expand-Concentrate
FIFO	: First-In First-Out
GFC	: Generic Flow Control

GN	: Gamma Network
HOL	: Head Of Line
IETF	: Internet Engineering Task Force
IP	: Internet Protocol
IPng	: Internet Protocol next generation
IPv4	: Internet Protocol version 4
IPv6	: Internet Protocol version 6
ITU-T	: International Telecommunications Unit-Telecommunications Standardization Sector
KN	: Kappa Network
LAN	: Local Area Network
LANE	: LAN Emulation
LFSR	: Linear Feedback Shift Register
MIN	: Multistage Interconnection Network
MISR	: Multiple In Shift Register
MPOA	: MultiProtocol Over ATM
MTTF	: Mean Time To Failure
NC	: No Controls
NMC	: Network Main Controller
ORA	: Output Response Analyzer
PO	: Pushout
QoS	: Quality of Service
RBP	: Restricted Backpressure
SE	: Switching Element
SIN	: Single-stage Interconnection Network
SRN	: Semi-Rearrangeably Nonblocking
STD	: Synchronous Time Division
RI	: Ring Interface
RTL	: Register Transfer Level
UPC	: Usage Parameter Control
URT	: Uniform Random Traffic
VCI	: Virtual Channel Identifier
VHDL	: Very high speed integrated circuit High Level Description Language
VPI	: Virtual Path Identifier

Chapter 1

Introduction

1.1 Background

Currently, the telecommunications industry is split between two protocols; namely the asynchronous transfer mode (ATM) and the internet protocol (IP). ATM has been identified by the International Telecommunications Unit - Telecommunications Standardization Sector (ITU-T) as a comprehensive switching technique that is capable of meeting the requirements of B-ISDN: such as high throughput, low switching delay, low packet loss probability, expandability, testability, fault tolerance, low cost, and ability to achieve broadcasting as well as multicasting. B-ISDN (broadband integrated service digital network) is the extension of the ISDN (previously defined for telephone and data transmission services). B-ISDN will be capable of providing all services (audio, video, and data transfer applications) in a unified fashion to various places at varying speeds. The advantages are ease of installation and maintenance, better user access and economical service, and flexibility in the introduction and evolution of services. ATM is also expected to provide communication services with negotiable quality of service (QoS) levels. IP was invented to be the network layer (layer 3 in the OSI model) protocol running on the top of Ethernet and Token Ring local area networks (LANs). The original versions of IP, such as IP version 4 (IPv4), are unreliable. Messages are dropped without retransmission and without warning the transmitting applications, i.e. it does not support QoS. A new IP standard has been proposed to support QoS and to solve the well known problem of address exhaustion. The new standard is known as IP next generation (IPng) or IPv6 [11].

Both the ATM and IP are packet switching protocols; however, there are major differences between them:

- IP carries variable-length packets, with lengths varying from 12 up to 64k octets; ATM carries fixed-length cell of 53 octets.
- ATM is a connection-oriented protocol, i.e. a path should be established for each connection; IP is a connectionless protocol.
- IP is generally datagram; ATM is switched-path.
- IP is designed for broadcast media and many of the features of the IP rely on this broadcast nature; ATM is fundamentally a point-to-point protocol, having to simulate a broadcast by using point-to-multipoint or multiple point-to-point connections.

In this chapter, we will not discuss the ATM or the IP protocols in detail. They are discussed extensively in the literature [12, 13, 14, 15]. Here we will only concentrate on the switching aspects in both protocols.

1.2 ATM Switching and IP Switching

ATM networks are based on the following concepts: 1) virtual circuits, 2) fixed-size packets (cells), 3) small packet size, 4) statistical multiplexing, and 5) integrated services. The key to providing integrated services is for connections of each service type to get a different QoS. This QoS is provided by the network resources such as end-systems, network controllers, and switch fabrics. With the diverse types of services expected in the ATM networks, the QoS requirements differ from one service to another. Some services are sensitive to delays, such as real time applications, and some other services are sensitive to cell loss, such as data transfer. Also, the sensitivity level differs from one end-user to another. The maximum cell rate, maximum cell delay, average cell delay, and cell loss probability are parameters used to characterize the QoS. If an end-system is requesting a certain level of QoS that the network resources can not furnish, then the network blocks this request. In some cases the network rejects a request of an end-system even if there are sufficient resources for that request [16]. This takes place when the request violates the initial QoS contract between the network and the end-system.

Congestion and flow control are among the most complex issues with the ATM technology. The reasons for this complexity are very evident [17]. Firstly, the diverse service classes that ATM promises to handle with acceptable QoS levels. Secondly, the ATM technology will appear in different network domains - LAN, MAN, and WAN - that have large differences in the field equipment such as speed and buffering. Thirdly, although the ATM switches can protect themselves from congestion by discarding cells, this may result in poorer overall throughput of the network especially for applications which employ encryption/compression where even losing one cell can lead to losing the whole message sent. There are nine mechanisms used by current ATM switches to handle congestion and flcw control. These mechanisms are [17]:

1. Connection Admission Control (CAC);

2. Usage Parameter Control (UPC);

3. Selective Cell Discarding;

4. Traffic Shaping;

5. Explicit Forward Congestion Indication (EFCI);

6. Resource Management Using Virtual Paths;

7. Frame Discard;

8. Generic Flow Control (GFC);

9. ABR Flow Control.

The above mechanisms reduce congestion in the network by either impeding the traffic that enters the network until the resources become available, or dropping low priority traffic and allow the higher priority traffic instead. The performance of the above mechanisms is greatly affected by the availability of the network resources.

The IP protocol is a routed protocol which can easily scale up. However, routing is not as efficient as the switching process. Accordingly, the current trend is towards integrating the IP routing protocol in the ATM switching networks. There are classical methods which are proposed by the Internet Engineering Task Force (IETF) and label-based IP switching methods proposed by the industry [11]. The classical methods approaches were criticized by the industry because they separated IP on the top of ATM without taking any advantages of the ATM features. This leads to replication of functions and complications of the network management. Examples for the classical methods are the LAN emulation (LANE) and the multiprotocol over ATM (MPOA). In the label-based IP switching methods, companies such as Cisco and Ipsilon are integrating the IP layer with the ATM switching layer. Examples for the label-based IP switching methods are the Tag switching and IP switching.

Recently, the new Gigabit switched Ethernet networks are offering a good alternative by aggressively increasing the bandwidth to the gigabit range. Such a development might lead to the old protocol stack using IPv6, which features QoS, running on the top of gigabit switched Ethernet network at the LAN level. Such high speeds with the next generation IP seems to be an easy migration from the legacy LAN systems.

From the above discussion, it appears that the industry it trending towards switching rather than routing. Switch fabrics are the core of the switching process. Switch fabrics are placed in switching nodes scattered all over the network. The operation of any switching mechanism is directly (or indirectly) influenced by the performance of the switch fabrics. Less efficient switch fabrics result in poor performance of the whole network due to delays and retransmission of the lost traffic. Efficient switch fabrics are characterized by their high throughput and low delay.

Figure 1.1 depicts a block diagram of a switch fabric. A switch fabric should feature some or all of the following functions:

- Cell buffering;
- Traffic concentration and multiplexing;
- Fault tolerance;
- Multicasting and broadcasting;
- Cell scheduling based on delay priorities;
- Selective cell discarding based on loss priorities;
- Congestion monitoring.

Indeed, building up a switch fabric which features all of the above functions is a challenging objective. Cell buffering is an essential function of the switch fabric, especially in case of congestion. Traffic concentration and multiplexing is a natural

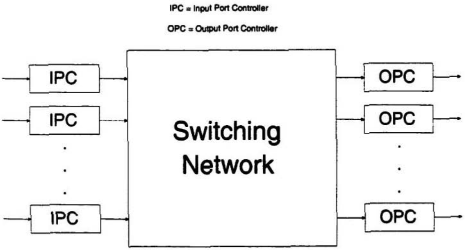


Figure 1.1: A block diagram of a switch fabric.

role of a switch fabric. A switch fabric concentrates and multiplexes the arriving traffic onto the OPCs. Fault tolerance has become a very important feature in today's communications equipment where the down time of the network is very expensive. A fault tolerance model is usually associated with the testability level of a system. Because the B-ISDN network will provide services such as video on-demand and video conferencing, multicasting and broadcasting are needed to support these services. Scheduling is necessary for the network to efficiently manipulate the different traffic types that arise from the various service classes. Cell discarding is essential in case of congestion. Switch fabrics drop cells that have lower priorities. For the success of the congestion and flow control mechanisms we discussed earlier, a switch fabric is required to monitor the amount of congestion it suffers and passes this to the network control plane. Hence, the congestion and flow control mechanisms take the proper decisions.

1.3 Motivation

In this dissertation, we provide a comprehensive study of a promising switching architecture called the Balanced Gamma (BG) network. With infinite buffering resources, the BG network has shown good performance in terms of throughput. average cell delay, and reliability. To continue studying the BG network, we further investigate the performance of the network with realistic buffering resources. Realistic buffer resources are comprised of finite queues. We investigate the performance of the network under different payloads containing uniform and non uniform traffic. Uniform random and bursty are the traffic types used. Several simulation experiments are carried out to measure the cell loss, cell average delay, and buffering requirements of the BG network. In addition, we pursued an analytical model under uniform random traffic to verify our simulation results. The performance of the network is compared with both an ideal network and the crossbar network. The performance results of the ideal network are used as an upper bound and it is shown how the BG network relates to that bound. The selection of the crossbar network was essential to prove the efficient performance of the BG network. The crossbar network is recognized by many researchers as a suitable candidate for ATM switching because of its good internal blocking characteristics.

The reliability is a measure of the system's ability to operate without failures during a specified period of time. It also measures the system's ability to tolerate faults. Previously, the reliability models of the BG network were established and showed that the reliability of the BG network outperforms other competitive networks. However, the network reliability model for the BG network was incomplete due its complicated nature. In this dissertation, we introduce a complete network reliability model of the BG network.

To emphasize the modularity and scalability of the BG network we decided to carry out a VLSI design for the network. We also wanted to prove that the BG network could be efficiently realized using the current available VLSI technologies. A 0.35 μm was the technology available to us during the design phase.

1.4 Thesis Organization

The thesis is divided into four parts. In the first part, which can be found in Chapter 2. we provide a survey of the switching architectures meant for broadband communications. The survey sheds light on the different classifications proposed by the researchers to classify the broadband architectures, emphasizing the differences and the similarities amongst them. In the survey, through examples we describe the advantages and the disadvantages of each class. We also discuss the recent techniques used to improve the performance the existing architectures.

In the second part, which can be found in Chapter 3, we briefly introduce the BG network and the previous efforts made to improve and simplify the network routing algorithm. We also emphasize our contribution in simplifying the routing algorithm and the necessary changes that had to be made to the network topology.

In the third part, which can be found in Chapter 4, we investigate the performance of the network with finite buffering resources in both the IPCs and the OPCs. We use both uniform and non uniform traffic loads. The load types we use are the uniform random traffic (URT) and bursty. The parameters used to measure the performance are the cell loss probability, maximum cell delay, average cell delay, input buffer requirements, and output buffer requirements. As we mentioned earlier, our investigation of the BG network will be compared with the performance of both the crossbar and ideal non blocking networks.

In the fourth part, which is composed of both Chapters 5 and 6, we introduce the VLSI design and the exact network reliability modelling for the BG network. The VLSI design is carried out using Synopsys CAD tool supported by the Canadian Microelectronics Corporation (CMC). We also use the 0.35 μm CMOS technology, also supported by CMC, to achieve a high speed design. Very high speed integrated circuit High Level Description Language (VHDL) is used to describe the design at the Register Transfer Level (RTL). We finally conclude our work in Chapter 7 and give some directions for future open problems in the area.

Chapter 2

Fast Packet Switching Networks

2.1 Introduction

Previously, the interest in fast packet switching networks was due to their use in fast parallel computing machines. Recently, more attention has been focused on these networks because of the evolving demands of broadband communications. Many classifications of these networks have been reported in the literature. In this chapter we shed some light on these classifications commenting on the similarities and differences amongst them. Our coverage will include the state-of-the-art of the architectures proposed for different subclasses showing the advantages and limitations of each. Since there is a great number of studies available in the literature that can not be covered in this chapter, we try only to focus on those architectures related to the scope of this dissertation.

2.2 Classifications of Packet Switching Networks

In the literature, many classifications of fast packet switching networks, also called switch fabrics, have been reported [1, 18, 19. 20, 21. 22]. Perhaps the classification model introduced in [1] is the simplest and the most comprehensive, because it has covered most of the well-known architectures. Figure 2.1 depicts the hierarchy of this classification. Similar, but less detailed. classifications are reported in [18] and [19]. Kyas [1] divides switching networks into two main classes, networks that use the time domain for switching, and networks that use the space domain for switching. In time division architectures, the physical resource is multiplexed among the input-output connections. based on discrete time slots. This physical resource can be a shared memory or a shared medium. Ring and bus topologies are typical examples for the shared medium case. In case of space division architectures, the connections are based on the availability of nonconflicting physical paths within the fabric. Figure 2.1 shows that the family of space division architectures has more subclasses than the time division architectures. We will discuss each one of these subclasses in more detail. stressing on the space division class because the proposed switch fabric in this dissertation belongs to that subclass.

Turner and Yamanaka [20] grouped the architectures, in general. into three major categories: 1) single stage systems, 2) buffered multistage systems, and 3) unbuffered multistage systems. This classification is more directed to multistage interconnection networks (MINs), which is a subclass of the space division class in Kyas's classification model. The reason is conceivably due to the growing interest in these networks. MINs have many advantageous features over other interconnection networks as we will explain later. Turner and Yamanaka inherently assumed that all switch fabrics are space division architectures, whether single stage or multistage, by including all the time division architectures within the single stage category. This assumption is acceptable because the traffic is relayed through one single shared medium in time division architectures. The classification also emphasizes the role of buffering in MINs.

Before we discuss the details of these architectures, we give a notation hint. Unless otherwise stated, all the architectures we discuss here are symmetric, i.e. the number of input ports equals the number of output ports and is represented as N.

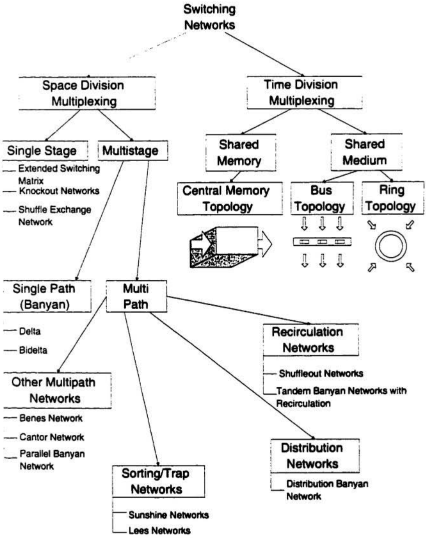


Figure 2.1: Classification of interconnection networks [1].

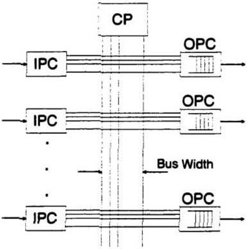


Figure 2.2: A basic bus-based switch fabric.

2.3 Time Division Switch Fabrics

2.3.1 Shared Medium Architectures

In fact, the first commercial ATM switch that hit the market was a bus-based architecture (ASX-100) provided by Fore Systems [23]. A typical simple bus-based architecture is shown in Figure 2.2. The IPCs do the ATM layer functions associated with checking for ATM header errors, VCI and VPI translation and synchronization of the arriving data stream to the internal switch timing. To relay the data to the OPCs. IPCs contend for access to the bus using one of a variety of bus-contention techniques. The control port (CP) plays the role of controlling the bus. Each OPC checks the destination addresses in each cell to decide whether to buffer this cell or not. Each OPC contains an internal buffer queue to hold the arriving cells. Obviously, the bus rate should be at least N times as fast as the rate of any individual IPC to alleviate the internal blocking problem, where N is the number of IPCs. The

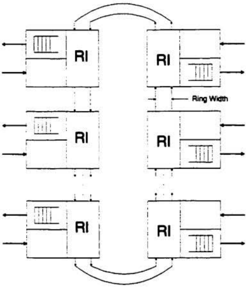


Figure 2.3: A basic ring-based switch fabric.

way to achieve such a rate is by increasing the width of the bus. For example, a system supporting 16 OC-3 links with internal bus rate of 40 MHz, requires a 64-bit bus width. Notice that as the number of the ports in the system increases, both the number of ports connected to the bus, as well as, the width of the bus must increase. This yields a quadratic growth characteristic, making it uneconomical to implement large switches. Another crucial problem with the bus systems is that as the number of I/O ports increases, the capacitive loading on the lines increases, reducing the internal bus rate.

A simplified ring-based system is depicted in Figure 2.3. The components of the system are similar to those in a bus-based system except that each pair of IPC and OPC are integrated in one unit and served by a ring interface (RI). The token-ring protocol is the most common protocol running the ring-based networks. Ring systems have the same quadratic growth problem as bus systems. However, they do not suffer from the capacitive loading problem. That is a ring system can run at faster rates than a bus system if both are implemented using the same technology. This leads to smaller ring width than wide bus width in the bus systems.

Based on the above discussion, we conclude that both ring and bus systems have pros and cons. Both have the quadratic growth characteristic as the number of ports increases. Bus systems suffer from capacitance loading but they enjoy the simplicity of implementing multicasting (we will discuss multicasting property later in more detail). On the contrary, ring systems do not suffer from capacitance loading because they are direct point-to-point systems, but implementation of multicasting is more complicated than bus systems. Additionally, in ring systems extra latency is added to pass the cells amongst the RIs. In general, we can state that shared medium systems are not the promising solution for future B-ISDN due to their uneconomical implementation.

2.3.2 Shared Memory Architectures

A multi I/O ports memory can represent a shared memory, also called shared buffer, switch. The IPCs write the arriving cells at the beginning of the switching cycle to the shared memory space and the OPCs read the cells that request them from the shared memory space. The OPCs identify the cells that request them with the aid of address holding queues (AHQs). The number of these queues is equal to the number of OPCs and each queue, which is associated with an OPC, holds the starting addresses of the cells in the shared memory that request this output. One bottleneck we can easily recognize in shared memory switches is that they should be centrally controlled to organize the process of reading/writing amongst I/O ports. Central control is one major obstacle towards building a large switch fabric. The main advantage of pure shared memory systems is that they have the lowest buffering requirements compared to any other switching architectures that have the same number of I/O ports. This is due to the complete sharing of the storage medium of the switch fabric, whereas in other architectures the buffering medium is partially or completely divided amongst system ports in the form of separate queues. Turner *et al.* [20] have discovered that shared memory systems reduce the memory buffering requirements by a factor of 5 to 8 less than shared medium systems. This is a crucial fact when the traffic load has a bursty nature. Another bottleneck is the memory access speed, which also declines as the memory size increases. Obviously, we have 2 N read/write operations have to be carried out in each switching cycle. That is, the memory access speed should be at least 2 N times as fast as any I/O port. This limit is twice the minimum limit of the bus rate in shared medium systems. That means for the previous example of the 16 OC-3 IPCs system, with internal bus rate of 40 MHz, we need a bus width of 128 for a pure shared memory system.

Shared memory systems suffer from similar problems we discussed above for shared medium systems, such as wide bus width and capacitance loading. Also, Yamanaka *et al.* [2] provided a quantitative discussion about the uneconomical implementation of the shared memory architectures. For a shared memory system of $2N$ input and output ports and total shared buffer size BN cells, the total memory space required for the AHQs in bits is given by [2]:

$$\text{AHQs size} = PBN^2 \log_2(BN), \tag{2.1}$$

where P is the number of service priorities. Figure 2.4 depicts the memory capacity requirements of the AHQs and the shared buffer. For a 32 × 32 switch with 1K cells buffers (when B = 32), no fewer than 320K bits are necessary in total for AHQs. Even with today's technology, this is too large to integrate into the same die

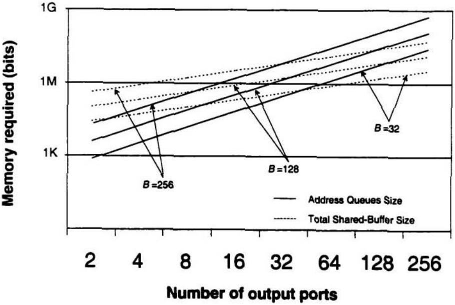


Figure 2.4: Memory capacity of AHQs versus shared buffer [2].

with the control parts of the system. Therefore, external RAMs or FIFOs have to be introduced, which often results in pin number bottleneck. The other interesting fact is that for only one service priority ($P = 1$), the memory capacity needed for the AHQs is greater than the shared buffer needed to store the cells themselves for network sizes > 32. One can visualize the added complexity if more than one service priority ($P > 1$) are provided. Several methods have been reported in the literature to manage the operation of the AHQs. A summary of these methods can be found in [24].

Perhaps, the PRELUDE architecture is the first trial to build a pure shared memory packet switch fabric [25]. The architecture of the 16 × 16 PRELUDE switch is an asynchronous version of the classical synchronous time division (STD) switch used in telephone networks [3]. The switch has gone through many phases of development. As early as 1986, a throughput of 16 × 280 Mb/s was achieved using emitter coupled logic (ECL) technology at the expense of a consumption of 400 W at 5 V. Progress in CMOS technology has led to the implementation of a second version of 24 CMOS chips in 1993 named COM16 [26]. The achieved throughput was 16 × 155.52 Mb/s while consumption was limited to 40 W at 5 V. The third version is a monochip COM16M that incorporate all the switch functions. Figure 2.5 depicts that architecture.

In [27], the authors reviewed the efforts by a research team at Bell Labs towards building an ATM switch. The group concluded that a pure shared memory switch has become a reality due to the advances in VLSI technology. The first prototype was an 8 × 8 switch fabric that had a 2K cells storage static RAM of 10-nsec access time using a 0.8 BiCMOS technology. The port speed for each I/O line of the switch fabric is 2.5 Gb/s making the total switch capacity of up to 20 Gbps. Further improvements were made to reduce the area of the switch and to increase the capacity up to 160 Gb/s for a

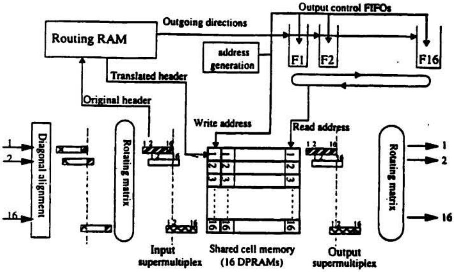


Figure 2.5: COM16M architecture [3].

switch fabric. However, the new modified switch fabric is not a pure shared memory architecture due to the above mentioned limitations, but it is a mixture of shared memory and space division architectures. Every eight consecutive output ports are grouped in one set and served by the initially developed pure 8 × 8 shared memory switch fabric. That is, there exist eight shared memory switch fabrics preceded by an expand-concentrate (EXCON) stage used to distribute the arriving traffic amongst the eight switch fabrics based on the requests made by the arriving cells. Generally, the switch performance is outstanding but the complexity of the switch is very obvious.

Another challenging shared memory switching module is proposed in [28] as well as a switch fabric based on that switching module. It is a clear example of how much complexity one contends with in designing centrally controlled architectures. The switch performance, although not provided, could be outstanding but a closer look at the switch architecture reveals that the cost/performance ratio is very high for this switch fabric. Additionally, the connections between the fabric central controller and different modules suggest that the implementation of the system has to be split over many boards making it a very difficult target for the system to attain high speeds.

It appears, due to the limitations we discussed above, researchers are tending towards hybrid systems. That is, networks are implemented as a mix of both shared memory and space division. Some newly reported examples that describe this trend can be found in [24] and [2]. In the subsequent sections we will discuss different buffering schemes adopted in space division architectures, showing that shared buffering scheme outperforms other buffering schemes.

2.4 Space Division Switch Fabrics

As mentioned earlier, the class of space division architectures has a larger collection when compared to the other class of time division architectures. As depicted in Figure 2.1, space division architectures can be divided into two main subclasses; namely single-stage interconnection networks (SINs) and multistage interconnection networks (MINs). Obviously, as both names of the subclasses indicate, architectures that belong to SINs are composed only of one stage, whereas architectures belonging to MINs are composed of multiple stages. In the next sections we discuss each subclass with the provision of some illustrative examples for each.

2.4.1 Single Stage Architectures

There are two well-known examples for SINs, the Crossbar (sometimes called Dot Matrix) switch and the Knockout switch. The crossbar switch is depicted in Figure 2.6. An SE is located at each crosspoint and takes one of two states, either cross state or bar state as illustrated in Figure 2.7. The crossbar switch outperforms several other switching networks. The main reason is that the switch does not suffer from any internal blocking. However, it suffers from output blocking because each main

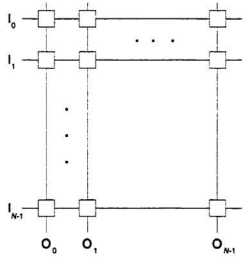


Figure 2.6: A crossbar switch fabric.

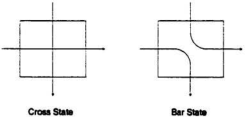


Figure 2.7: Crossbar SE states.

output can not accept more than one cell in any switching cycle. Furthermore, the switch has two main disadvantages, which are intensified as the size of the network (N) increases. Firstly, the hardware complexity of the switch is $O(N^2)$, which is highly complex when compared to other space division architectures. However, recently Woo [29] has proposed a new design that reduced the switch complexity to $O(N\sqrt{N})$ by adopting the theory of *finite projective planes*. Although the new idea has reduced the complexity by a factor of $\sqrt{N}$, the hardware complexity of the newly designed SEs to implement the new proposed switching protocol has significantly increased. Secondly, there is lack of fairness amongst the input links of the network. The problem takes place because the internal SEs may give priority to the input ports of the network in a descending or ascending order, depending on the routing decisions made within the SEs. A randomizing mechanism may be used to alleviate the problem but at the expense of larger hardware complexity of the SEs. In [30], under uniform random traffic a theoretical upper limit for a maximum throughput of 0.5858 has been reached for the crossbar switch of infinite size ($N = \infty$) and infinite input buffering. Further investigations of the crossbar performance under different buffer strategies can be found in [31, 32]. To conclude, the crossbar switch could be considered the proper choice for the circuit switching technique, but it has lost attraction with the new demands of the broadband packet switching where scenarios of output traffic concentration exist. In [20], a survey of different methods used to improve the crossbar network performance -such as introducing buffering within the crosspoints, having multiple buffer queues at each input port and speeding up the fabric switching- is conducted. Despite the complexity of the crossbar switch, recent implementations that are based on crossbar architecture can be found in [33] and [34].

The architecture of the knockout switch is depicted in Figure 2.8. Each output link O_i is preceded by an N:L concentrator. The concentrator has three main parts,

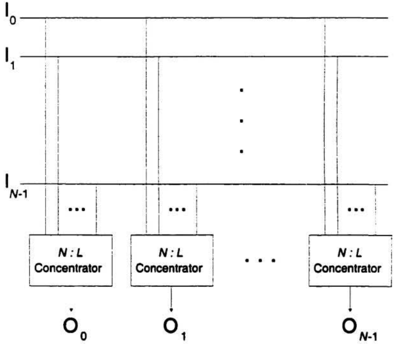


Figure 2.8: Knockout switch architecture.

1) N address filters, 2) N:L multiplexer, and 3) a buffer space. The address filters pass only the proper cells and reject all other cells that request other output ports. The concentrators confine the arriving cells in a parallel stream of L cells. That is if more than L cells request the same output port, the multiplexer has to reject some of the excess cells, hence the name *knockout*. Finally, the buffer stores the arriving cells. If $L = N$, then the switch is called as a *perfect* switch. The reason is that no sort of blocking is experienced in the network. This does not mean that the network has a perfect performance, because the output queuing problem still exists. When the incoming traffic is more bursty, the output buffer needs to be larger to prevent buffer overflow. This problem, which we refer to as the output queuing problem, is a classical queuing theory issue. The network is sometimes called as a "*Disjoint-path*" network, because there is no intersection amongst input/output connections. Obviously, the knockout switch suffers from the capacitance loading problem discussed earlier for bus and shared memory architectures. In fact, the performance of the knockout switch is very similar to the performance of shared memory architectures because the knockout switch does not suffer from any internal blocking. Note that the complexity of the concentrator grows drastically as N or L increases. In [5], a survey of some tradeoffs to implement the knockout switch is conducted. Throughout this dissertation, we selected both the crossbar and knockout switches to compare their performances with the architecture we propose. We call the version of the knockout switch we use a *perfect* switch because we set $L = N$.

At this stage, the different blocking definitions for interconnection networks are worth mentioning. These definitions describe the ability of any switching network to make different permutation connections. Clearly, a nonblocking $N \times N$ can realize $N!$ permutations. The definitions are as follows [5]:

- A network is said to be *nonblocking in the strict sense or strictly nonblocking* if it is capable of immediately establishing a connection between any input-output pair without interference from any arbitrary existing connections.

- A network is said to be *nonblocking in the wide sense or wide-sense nonblocking* if any desired connection between an input-output pair can be established immediately, provided that the existing connections have been inserted using some routing algorithm. If the algorithm is not followed, some attempted connections may get blocked.

- A network is said to be *rearrangeable or rearrangeably nonblocking* if a desired connection between any input-output pair can be established if one or more of the existing connections are rerouted or rearranged.

- A network is said to be *blocking* if some connection sets can prevent some desired connections from being established.

Although, these definitions are standard and used by many researchers to classify switching architectures, they may provide a misleading picture about the performance of a switching architecture. The fact is, permutation traffic is not the traffic type anticipated in the broadband communication systems. In the broadband traffic scenarios, multicast and broadcast connections, as well as output concentrations, are expected. These traffic loads degrade the performance of many existing switching architectures significantly. A typical example, as we will see later, is the *Batcher banyan* switch.

2.4.2 Multistage Interconnection Networks

Recently, the interest in MINs has grown very fast. This is because of two main reasons. Firstly, in MINs no central control is needed to supervise the switching process as the case in shared memory and shared medium systems. That is, the control is distributed amongst the SEs. This property is called self-routing. Secondly, the architectures of these networks are modular making them attractive for VLSI implementation. We first discuss the single path and then the multipath MINS in the next two sections.

2.4.2.1 Single Path MINs

Goke and *Lipovski* [35] defined that, a network with a unique path from each input to each output is called a ***banyan*** network. Also, another definition by ***Patel*** [36] is:

> A *delta* network is an $a^n \times b^n$ switching network with n stages, consisting of $a \times b$ crossbar modules. The link pattern between stages is such that there exists a ***unique*** path of constant length from any source to any destination.

Obviously, the two definitions are close and they are used interchangeably in the literature. In general, we will limit our discussion only for rectangular MINs. A rectangular MIN is of size $N \times N$. Several well-known MINs belong to delta class such as banyan [35], baseline [37], reverse baseline [37], omega [38], modified data manipulator [37], and indirect binary n-cube [39]. Some of these networks are shown in Figure 2.9. A rectangular banyan network based on 2×2 SEs has $log_2 N$ stages. Each stage has $\frac{N}{2}$ SEs. In addition to the above mentioned advantages of MINs, banyan networks based on 2×2 SEs enjoy the privilege of the simple structure of their SEs. A 2×2 SE is very simple to implement because the number of decisions needed to be taken are smaller compared to higher order SEs.

In [40], it has been shown that the performance under uniform random traffic is the same for all banyan configurations. Despite all the advantages, the performance of a banyan network is extremely poor even under simple traffic loads, such as uniform

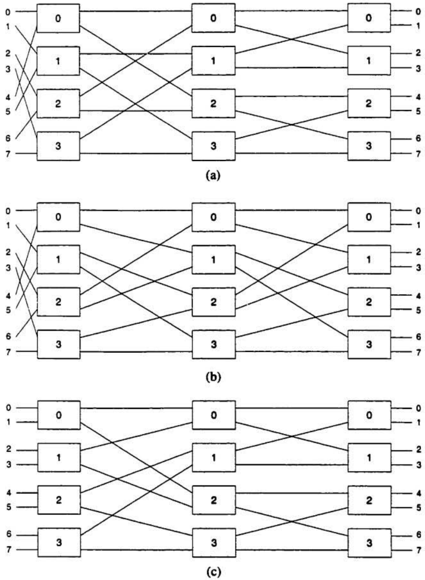


Figure 2.9: Different banyan network topologies (a) banyan (b) omega (c) baseline.

random traffic. Several methods were proposed to improve the performance but this is achieved at the expense of the network complexity. The most well-known method is using buffers. We will provide a detailed discussion of the buffering techniques in Chapter 4.

2.4.2.2 Multipath MINs

The subclass of multipath MINs has a huge collection, because all single path MINs are inherently included. Later we will provide discussions about some methods used to build multipath MINs from single path ones. Prior to that, we first discuss the structures of standalone multipath MINs.

Clos network

The Clos network [41] has been in the research focus for decades. The reason is that it is a nonblocking network with fewer crossconnects than the crossbar switch. This fact made the Clos network a strong candidate for circuit switching networks. However, the Clos switch was also considered for packet switched networks as in Memphis switch built for IBM GF-11 parallel computer [42]. The Clos network is a three-stage network that can be built in asymmetric or symmetric fashion. The first stage of the network has r_1 $(n_1 \times m)$ SEs, the middle stage has m $(r_1 \times r_2)$ SEs, and the third stage has r_2 $(m \times n_2)$ SEs. In a symmetric network $r_1 = r_2 = r$ and $n_1 = n_2 = n$. Figure 2.10 depicts the architecture of a symmetric Clos network. Clos networks can be designed to either operate in rearrangeable nonblocking mode or nonblocking mode. It has been shown for a symmetric Clos network [43]:

- if $m \geq n$ then the network can operate in rearrangeable nonblocking mode,
- if $m \geq \lfloor \frac{3n}{2} \rfloor$ then the network can operate in the wide-sense nonblocking mode,
- and if $m \geq 2n - 1$ then the network can operate in the strictly nonblocking

mode.

Obviously, the hardware requirement for a Clos network to operate in a strictly nonblocking mode or wide-sense nonblocking mode is higher than that to operate in a rearrangeable nonblocking mode. Accordingly, a symmetric *semi-rearrangeably nonblocking* (SRN) Clos network has been provided [44]. Similar to a nonblocking network, an SRN network does not allow any arrangement of the existing connections in order to establish a new connection; however, unlike a nonblocking switch, an SRN switch allows at most one rearrangement to take place when an existing connection is disconnected. Thus the overhead of realizing a new connection under SRN mode of operation is the same as that under the nonblocking operation. However, the overhead of disconnection in the SRN mode is slightly higher than that in the nonblocking mode of operation. Nevertheless, this increase in the overhead is justified because the hardware resources required for SRN are much lower than those required for nonblocking mode of operation [44]. Recently, a similar approach was proposed for asymmetric Clos networks [45].

In [46], an optical implementation and a performance analysis of a symmetric Clos network were proposed. The performance was studied under uniform random traffic. It was discovered that the performance is acceptable under low traffic loads, but as the load increases the amount of internal buffering increases. One drawback of internal buffering within MINs is the emergence of the so called *out-of-sequence* problem, where cells reach the destinations not in the order they arrived at the input ports. In the literature, there are two schemes used to control switching in Clos network to overcome the out-of-sequence problem but at high cost of complexity. The first scheme, which is used in [45], is called *multirate circuit switching* [47]. The second is a dynamic routing scheme called *cell switching* [48]. In the former scheme, a path

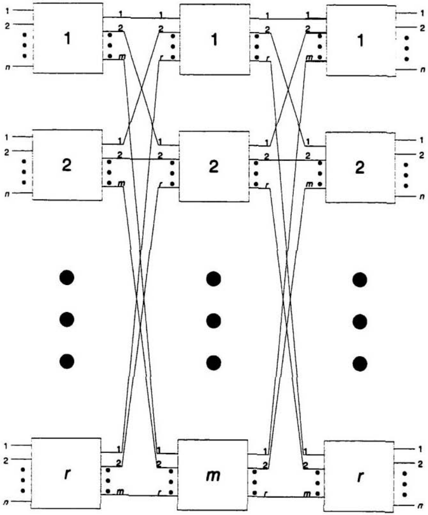


Figure 2.10: Architecture of the Clos network.

is established for each connection such that the peak capacity is reserved along the path and it is sufficient to carry the connection at any time. Clearly, this scheme will result in low utilization of the network capacity because it does not take advantage of the statistical multiplexing offered by the ATM protocol. But, it provides a guarantee of QoS for each connection. The latter scheme is more dynamic, thus it has better utilization, but extra overhead is needed to overcome the out-of-sequence problem. A recent scheme, which lies in the region between both previous schemes, has been proposed in [49]. Obviously, there are too many concerns that have to be taken into account when designing a switch based on the Clos network. These concerns result in complicated implementation of the switch and high overhead processing time, that makes it difficult to implement switches with large sizes based on the Clos network. Another recent optical implementation of an architecture based on Clos network can be found in [50].

Kappa Network

Kappa network (KN) [4] is a variation of the Gamma network (GN) [51]. Both GN and KN were proposed as fault-tolerant MINs. The GN is derived from the modified baseline network. Both KN and GN have $\log_2 N + 1$ stages, but with different SE sizes. The first stage of the GN has N 1×3 SEs, whereas the first stage of the KN is composed of N 1×4 SEs. Each middle stage in the GN is composed of N 3×3 SEs, whereas it is composed of N 4×4 SEs in the KN. The last stage of the GN has N 3×1 SEs, whereas the last stage of the KN has N 4×1 SEs. Figure 2.11 depicts the architecture of an 8×8 Kappa network. If we remove the links denoted by 'A' in Figure 2.11, we obtain the GN. Each link in the Kappa network has an alternate one. In case of a fault, the traffic is directed to the alternate link. Obviously, the family of banyan networks does not have any fault tolerance features. However, there are several methods to improve their fault tolerance properties by replicating the network

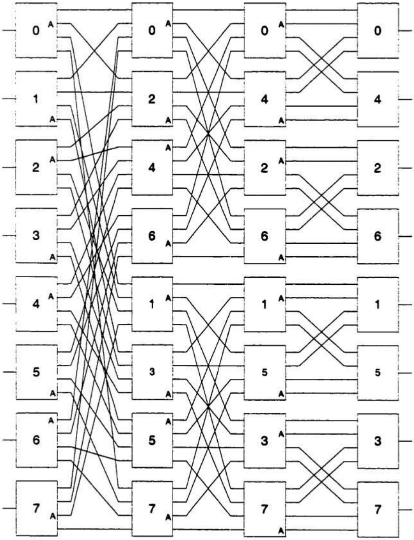


Figure 2.11: Architecture of an 8 × 8 Kappa network [4].

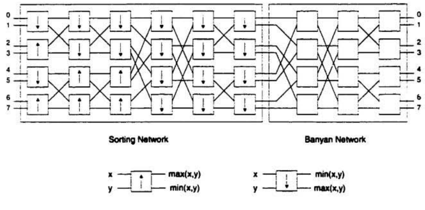


Figure 2.12: Architecture of an 8 × 8 BB network [5].

or dilation of the links. We will explain these concepts in the following sections.

Batcher banyan Network

Banyan networks become nonblocking under permutation traffic if the arriving cells are sorted in ascending or descending order; provided that no gaps exist between active inputs [52]. A Batcher banyan (BB) network is composed of a sorting (Batcher) network followed by a banyan network. Figure 2.12 depicts the architecture of an 8×8 BB network. The BB network is a multipath network because the path from any source to any destination depends on the sorting order in the sorting network. The sorting network [53] has $\frac{log_2N+1}{2}log_2N$ stages, with $N/2$ sorting elements in each stage. In total, $\frac{N}{2}(\frac{log_2N+1}{2}log_2N)$ sorting elements and $\frac{N}{2}log_2N$ SEs exist in the BB network. Obviously, the BB network has less crosspoints for large N if compared with the crossbar switch, however it is an expensive alternative to improve the performance of the banyan network. Despite that complexity, several switches that have been reported in the literature are based on the BB network [5]: Starlite switch, Sunshine switch , 3-phase BB switch, and 2-phase BB switch. Although the performance of the BB network is outstanding (100% throughput) under permutation traffic, it has poor performance under other traffic loads which are closer to the real ATM traffic. This can be rationalized due to the limited performance of the banyan network which constitutes the output part of the BB network. It is worth mentioning at this step that permutation traffic, which is used as the basis for defining the blocking properties of the different network architectures, does not really represent the real ATM traffic which is expected to be bursty and have multicast properties [54]. That is why it was reported in [55] that the BB network is more popular in the research communities than for commercial applications.

Extended Banyan Networks

Beneš [43] first introduced the concept of extended networks, hence these networks are called Beneš networks. The Beneš network is another example for a multipath network obtained from the banyan network. The network has $2log_2N-1$ stages, and is composed of two halves. Each half is a mirror image of the other as shown in Figure 2.13. The first half, which is composed of the first $log_2N - 1$ stages, is called the *distribution* network and the second half, which is composed of the last log_2N stages, is called the *routing* network. Notice that we can still obtain a multipath network even with a distribution network that has number of stages less than $log_2N - 1$. However, such networks have poorer nonblocking properties when compared to the generic Beneš network. Beneš has proved that such networks are rearrangeably nonblocking [43].

Beneš networks suffer from two drawbacks. Firstly, the control is centralized in the distribution network, which leads to losing the distributed control advantage that MINs enjoy. Secondly, the architecture does not overcome the out-of-sequence problem encountered if the SEs contain buffers because it is a multipath architecture. Recently De Marco and Pattavina [56] have defined a new distributed control algorithm, which is suitable for ATM cell switching, to control the switching operation in

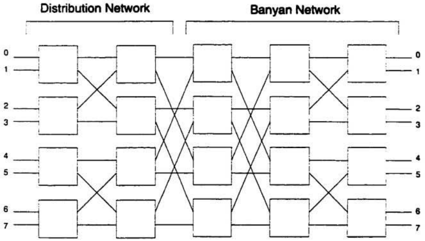


Figure 2.13: The architecture of an 8 × 8 Beneš network.

the extended banyan network. Despite this improvement, the complexity of the SEs in the distribution network increases drastically. This is caused by the complicated algorithm implemented in each SE. The functions of that algorithm include gathering information about the buffers in the same pool (see [56]), manipulating this information, and then taking the routing decision. Additionally, the output port controllers have to guarantee the sequence of the arriving cells because this algorithm does not obviate the out of sequence problem if the network is internally buffered. This scenario has to be repeated in every switching cycle. Indeed, the extended banyan network is attractive for circuit switching systems, as the case for Clos network, but is less efficient when used in packet switching systems.

Dilated Networks

A dilated banyan network has the same topology and the same number of SEs as the banyan network except that each link in the network is replicated many times. For example, the building SE for a 2-dilated banyan is shown in Figure 2.14. The up-

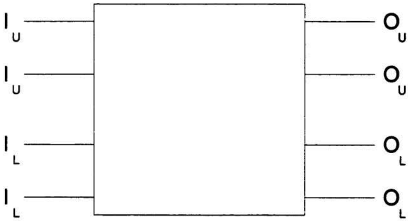


Figure 2.14: A switching element in a 2-dilated banyan network.

per/lower inputs(outputs) are connected to the same SE in the previous(next) stage. Importantly, we should not mix the network dilation with increasing the internal bandwidth of the network. The latter is similar to using wide bus for speeding up, discussed earlier in section 2.3, whereas in the former the number of ports per SE is increased. In fact, both concepts could be adopted in the same architecture. Note that the concept of dilation is not restricted only to the banyan networks, but indeed, it can be adopted with any other MIN architecture.

In the literature, the idea of dilated networks has been studied [57, 58]. As would be expected, The performance is shown to be better than the regular banyan network. That is because of the increased number of paths that a cell can go through. In [59], Alimuddin *et al.* have discovered that using fixed dilation, i.e., using the same dilation degree, for all SEs result in underutilization in the initial stages and overutilization in the last stages. They have proposed a structure where the degree of dilation is not the same in all stages. It increases towards the last stages of the network. That way

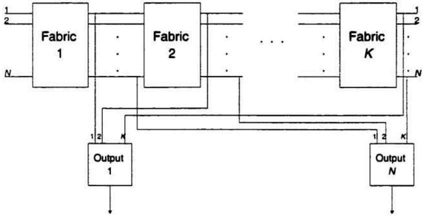


Figure 2.15: The tandem architecure.

scenarios of output port contention can be relaxed with the availability of more paths. Finally, we should be aware of the increased hardware complexity when incorporating dilation.

Tandem Networks

The first tandem architecture was first introduced by Tobagi *et al.* [60], and it was based on the banyan network. A tandem structure is composed of K fabrics connected in cascade as shown in 2.15. Each fabric can belong to any of the switching architectures we discussed above. The arriving cells are switched through fabric 1. If an internal conflict exists, some cells win contention and the others are routed to the other unused output links. After the first fabric, cells are checked and those have not reached their destinations are sent to fabric 2 and the other successful cells are received by the output ports. The process is repeated till reaching fabric K and then unsuccessful cells are dropped, or in some other architectures they could be recirculated back. Note that switching is taking place in all fabrics simultaneously, which may lead to the out-of-sequence problem. For example, if a cell loses contention in consecutive stages and the cells arriving later in the same logical connection win contention, then cells will arrive out-of-sequence at the destination.

The performance of the tandem banyan is definitely better than an ordinary banyan network as K increases. However, a tandem architecture is a very expensive solution especially if a low cell loss ratio and minimum number of out-of-sequence cells are two objectives. Because these two objectives translate to higher K and a complicated resequencing mechanism at the main output ports controllers of the network. The authors in [60] reported that, on average, the tandem banyan network of size $N = 32$ with recirculation needs at least 4 fabrics under uniform random traffic.

Parallel and Parallel–Like Networks

In parallel banyan networks, also called replicated banyan networks, there are K networks, each called a *plane*, connected in parallel. Each plane can be based on any MIN or switching architecture we discussed above. The traffic is divided amongst these K planes and then merged at the output ports. Many approaches can be followed to divide the arriving traffic. In the first approach, the arriving traffic is divided randomly with equal probabilities and each plane independently routes its share of the traffic to the output ports [61]. The second approach is to operate in a pipelined fashion [62], where each switching cycle is divided into K overlapped phases. In each phase the headers of the cells at the head of line (HOL) are first routed and successful headers that reach their destinations are positively acknowledged by the output ports. These positive acknowledgments are sent back to the input ports. This part of any phase is called the *reservation cycle*. Now the input ports react by relaying the body of these positively acknowledged cells through the plane which is corresponding to the current phase. Once the cells start to get relayed they are no longer at the HOL in the input ports. That is the input ports immediately start the reservation cycle of the next phase. The process is shown in Figure 2.16.

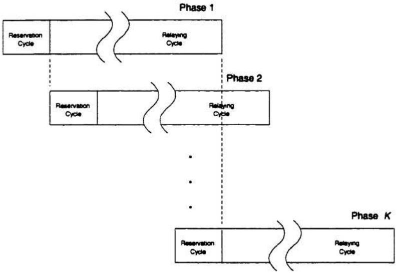


Figure 2.16: Timing sequence of the different pipeline phases.

In the first approach, the reservation in all planes is carried out at the same time. This is because each plane operates independently of all others. Whereas, in the second approach, the reservation cycles of all planes can be carried out in one separate unit (called control plane in [62]), and the routing decision can be passed to each plane. Notice the end of the reservation cycle of phase K is synchronized with the start of the reservation cycle of phase 1 to guarantee efficient utilization of the control plane. A structure based on the second approach is shown in Figure 2.17.

There are three advantages of the second approach over the first one. Firstly, the structure of any of the data planes is simple if compared with the structure of any of the parallel planes in the first approach because the data planes are not required to take any routing decisions. Secondly, the control plane can run at lower speeds than the data planes, depending on the number of data planes used. Table 2.1 illustrates

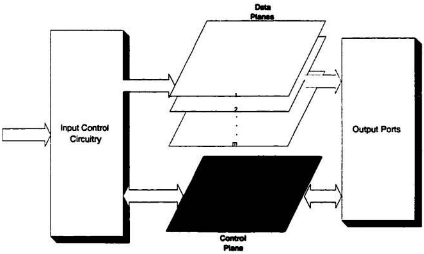


Figure 2.17: The pipeline architecture.

the theoretical maximum speed reduction ratio for the control plane of a pipeline structure. Obviously, as the speed reduction ratio decreases as the network size N or the number of data planes K increases. Thirdly, the second approach has better utilization of the data planes because no cells are lost due to internal blocking. Despite these advantages, the pipeline structure suffers from two main problems, which can be considered as advantages for the first approach. Firstly, the poor fault tolerance

N	K	Speed reduction ratio
32	3	8.8
64	3	7.4
128	4	4.8
256	4	4.24
512	4	3.78
1024	5	2.7

Table 2.1: Control plane speed reduction ratio [10]

properties of the network, because if a SE becomes faulty in the control plane, all other corresponding SEs in the data planes will be out of operation. To improve the fault tolerance properties Cheng *et al.* [63], proposed that each parallel plane takes its own routing decision instead of depending on one plane, but at the expense of extra hardware added to each plane. Secondly, the relaying cycles in the parallel planes are not synchronized, which demands a more complicated algorithm in the output ports to write to the buffers.

2.5 Summary

In this chapter, we introduced an overview of the different switching architectures proposed in the literature. We discussed the differences and similarities, as well as the appropriateness, of the most popular classifications used to classify these switching architectures. We also provided an example for each subclass in these classification hierarchies. Each example has demonstrated the advantages and disadvantages of each subclass. Additionally, we showed that the subclass of multipath MINs has the biggest collection when compared to other subclasses due to their modular, scalable and efficient performance. In the next chapter we will discuss the architecture we are proposing in this dissertation, which is a multipath MIN.

Chapter 3

Balanced Gamma Network

3.1 Introduction

The BG network is a promising MIN for broadband fast packet switching. The BG network offers an outstanding performance when compared with other well-known networks, such as the crossbar and the Batcher Banyan networks. In this chapter, we introduce a historical background as well as some new proposed modifications to the BG network. These new modifications concern both the routing algorithm and the network topology. We also provide some examples to describe the routing algorithm currently employed in the BG network. In addition, we discuss some preferred strategies adopted in the new routing algorithm.

3.2 Historical Background

3.2.1 Topology

The BG network was first reported in [64]. This network has a similar structure as the Kappa network [4], except in the last stage, the 4 to 1 concentrator at each output port of the kappa network is replaced by a buffer which is capable of receiving up to 4 cells in one switching cycle. The BG network has $n+1$ stages where $n = \log_2 N$. The first stage has 1×4 SEs. Each of the following $n-1$ stages have 4×4 crossbar

SEs. The last stage is a buffer stage as mentioned earlier. Each stage has N SEs numbered from 0 to $N-1$. Figure 3.1 depicts the initial structure of an 8×8 BG network.

Each SE in the BG network is addressed as $SE_{i,j}$ $(0 \leq i < N, 0 \leq j < n)$, where i is the number of the SE in stage j. Each SE has four output links numbered from 0 to 3 and indicated by $OL_0, \ldots, OL_3$. The output links of $SE_{i,j}$ are connected as follows: OL_0 is connected to one of the inputs of the $(i-2^j)^{th}$ SE in the next stage $(SE_{i-2^j,j+1})$, OL_1 is connected to one of the inputs of the $(i)^{th}$ SE in the next stage $(SE_{i,j+1})$, OL_2 is connected to one of the inputs of the $(i+2^j)^{th}$ SE in the next stage $(SE_{i+2^j,j+1})$, and OL_3 is connected to one of the inputs of the $(i+2^{j+1})^{th}$ SE in the next stage $(SE_{i+2^{j+1},j+1})$. In the next section, we discuss the routing algorithms proposed for the BG network.

3.2.2 Routing Algorithm

The BG network originally used a distance tag algorithm [64]. In that algorithm, a routing tag is defined by the input port controllers for each arriving cell. This tag is given by $(D-S)$ mod N where S denotes the input port the cell originating from and D is the cell destination. The SEs interpret the bits of the routing tag in a reverse order; that is, the SEs of $Stage_0$ use the least significant bit for routing. If the routing bit is '0', then the arriving cell is routed through OL_1. If the routing bit is '1', then the arriving cell is routed to OL_2. As there are four input links coming into each SE, it is quite likely that, in a given cycle, more than one packet may arrive at the same SE with the same routing bit (either 0 or 1). In case of no priority assigned to any cells, if three or more cells are having the same routing bit value, two are randomly selected and the rest are dropped. If the two selected cells have the routing bit value of '0', then one of them is routed to OL_3. If the two selected cells have the routing

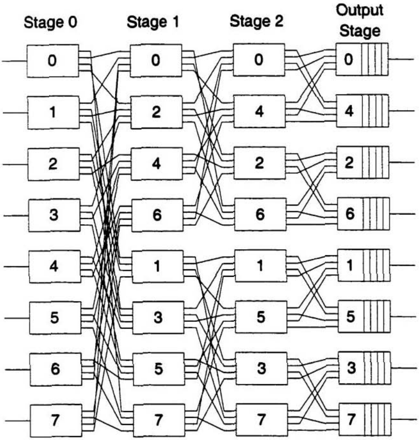


Figure 3.1: Initial BG structure.

bit value of '1', then one of them is routed to OL_2 and the other is routed to OL_0. However, the routing tag has to be modified for these cells directed to OL_0 and OL_3.

To reduce the overhead needed to generate the routing tags and modifying them in the above algorithm, another routing algorithm was proposed in [65]. The new algorithm is called reverse destination tag. The tag is only the destination address D in binary, viz. $d_{n-1}d_{n-2}\ldots d_0$. That way, the process needed to generate the tag at the input ports is not as complex as it is in the above algorithm. In this algorithm, the SEs also interpret the tag in a reverse order. Nonetheless, another parameter α has to be defined within each SE and used to take the routing decision. This parameter α is given by:

$$\alpha = \lfloor \frac{i}{2^j} \rfloor \bmod 2. \tag{3.1}$$

If ($d_j \oplus \alpha = 0$) for an arriving cell to $SE_{i,j}$, then $SE_{i,j}$ routes this cell through OL_1 or OL_3, else $SE_{i,j}$ routes this cell through OL_2 or OL_0. If more than two cells have the same routing bit tags, then two are selected randomly for routing and the rest are dropped. Notice, in this algorithm the SEs do not need to modify the routing tags for the cells going through OL_3 and OL_0.

3.3 Balanced Gamma New Structure

In this dissertation, we introduce a new routing algorithm that reduces the routing decision within the SEs to only checking the value of the routing tag bit. In the new algorithm, which uses the reverse destination routing tag, if the value of the routing tag bit is 0, then the arriving cell is routed through OL_0 or OL_1, else the arriving cell is routed through OL_2 or OL_3. We do not need to check for a parameter like α or modify the routing tag as the case in the above two routing algorithms. However, we ought to change the network topology to implement this new algorithm, but without affecting the network modularity enjoyed by the previous topology described above. The new topology is described in Appendix A and depicted in Figure 3.2. A typical connection, shown in thick line, between input port 3 and output port 6 in Figure 3.2 illustrates the new routing algorithm. In this connection, the routing tag is $6 = (110)_2$, then the SE in $Stage_0$ routes the cell to OL_0 because the routing bit is 0, whereas the SEs in the subsequent stages route the cell through OL_2 because the routing bit is 1.

Contrary to the topology used in the routing algorithms introduced in section 3.2.2, every output link in each SE is connected to a particular input link of the SE in the next stage. This allows us to treat different service priority cells by appropriately confining all the high priority traffic to specific output links. In fact, we decided to route high priority cells through OL_0 or OL_2 if the corresponding routing tag bit is a 0 or a 1, respectively. Consequently, the high priority cells will be confined to IL_0 and IL_2. However, the other links in the network can also carry high priority cells. In case of more than one cell are arriving to the same SE and having the same routing bit value, the top two high priority cells will win contention and others will be dropped. In the example of Figure 3.3, three arriving cells ('b', 'c', 'd') have a routing bit = '0' with cells 'b' and 'd' having high priority, and cell 'c' having low priority. Also there is one arriving cell ('a') having routing bit = '1' and high priority. Cells 'b' and 'd' are routed through OL_0 and OL_1, respectively, and cell 'c' is dropped. Cell 'a' is routed through OL_2 and an IDLE cell is routed through OL_3.

Also, we decided to adopt a deterministic routing algorithm rather than a random one. We found out this is better for two reasons. Firstly, the architecture of the SE is less complicated because no random number generator unit is existing. Secondly, the routing algorithm is simpler because no random number is checked. All the design issues will be discussed later in Chapter 5. Appendix B provides the details of the

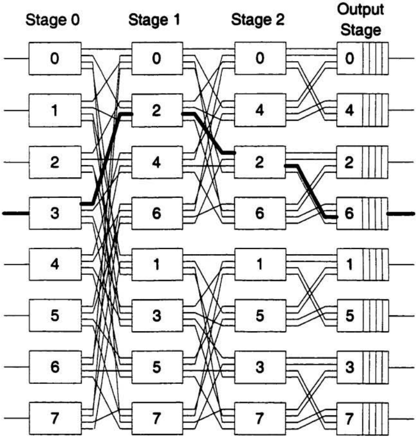


Figure 3.2: New BG structure.

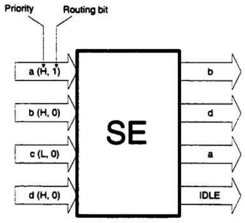


Figure 3.3: A SE routing decision example.

new deterministic routing algorithm.

3.4 Summary

In this chapter we have discussed the old and new architectures of the BG network. We have provided a historical background of the BG network previous architectures. Previously, two routing algorithms were proposed for the BG network. However, in this dissertation we proposed a new routing algorithm which is simpler than the previous ones. The new routing algorithm necessitated some modifications to the topology of the network. These new modifications did not affect the network complexity or modularity. Also, the proposed routing algorithm can handle two levels of priority cells. It tries to confine the high priority cells to specific internal links. In chapter 6 we discuss fault tolerance properties of the BG network in more detail.

Chapter 4

Performance Under Uniform and Non Uniform Traffic

4.1 Introduction

In this chapter, we investigate the performance of the BG network with finite buffering resources in both the IPCs and the OPCs. We use both uniform and non-uniform traffic loads. The load types we use are URT and bursty. The parameters used to measure the performance are the cell loss ratio, maximum cell delay, average cell delay, input buffer requirements, and output buffer requirements. The reader should be aware of the fact that both the throughput (TP) and cell loss ratio are related by:

$$TP = 1 - \text{cell loss ratio} \tag{4.1}$$

As mentioned earlier, the results of the BG network will be compared with the performance of both the crossbar and ideal non blocking networks. We first provide an overview of the buffering techniques that are used to enhance the performance of switch fabrics. Next, we discuss the performance under URT with the introduction of an analytical model for the pipelined BG network. The next section discusses the performance of the three architectures under different bursty traffic loads. We conclude this chapter by investigating the performance under various loads of non-uniform traffic.

4.2 Buffering Strategies

Due to the bursty nature of the traffic loads in broadband packet switching systems, buffering can not be avoided even when using an ideal nonblocking switch fabric. An example for this ideal nonblocking switch fabric is the *knockout* switch with $N = L$. This ideal fabric, in any switching cycle, is capable of fulfilling all the requests issued by the arriving cells at the input ports of the fabric. However, only one cell can depart from any of the output ports in a switching cycle. Accordingly, buffering should be adopted, otherwise the switch will suffer from cell loss. Notice that, buffering can not totally eliminate cell loss, but it can reduce it to acceptable levels. Adding buffers is not a penalty-free solution towards minimizing cell loss, simply because of two reasons. Firstly, buffering requires adding extra hardware to the system in the form of memory elements and control circuitries. We know that the speed of memory degenerates as the size increases, placing a limit on the maximum memory size that could be used. However, the speed of the storage system can be increased by bit slicing [27] which is an expensive solution. Secondly, cells which are stored within the buffers suffer from delay. In fact, the larger the buffer sizes the bigger is the delay, which imposes another limit on the buffer size used. In general, some applications are very sensitive to delays such as real time multimedia applications. Both peak and average cell delay values are two parameters used to negotiate the QoS during connection establishment in the broadband communication systems. Notice that both cell loss and cell delay are working against each other. For example, if we try to improve the level of cell loss, which is normally achieved by increasing buffer sizes, we end up by experiencing poor levels of cell delay, and vice versa.

Buffering strategies take different configurations depending on the switch architecture. We study the buffering strategies used in space division architectures in the next section.

4.2.1 Input/Output Buffering

A switch fabric can be pure-input buffered, pure-output buffered, or input-output buffered as depicted in Figure 4.1. In the pure-input buffering strategy, a buffering mechanism is located at each input line of the fabric. In each switching cycle, the fabric routes the cells at the HOL in the input buffers to the output ports. The input buffers can then operate in either a lossy mode or a backpressure mode. In the lossy mode, the input ports remove the unsuccessful routed cells from the input buffers and bring the cells at the next locations to the HOL to participate in the next switching cycle. While, in the backpressure mode the input ports are acknowledged, through an acknowledging mechanism that is distributed all over the fabric, with the successful routed cells. Based on the received acknowledgments, the input ports remove only the successful routed cells from the input buffers and keep the unsuccessful ones. It is obvious that a fabric operating in a lossy mode has a simpler hardware complexity compared to a fabric operating in the backpressure mode. This is due to the existence of the backpressure mechanism distributed all over the switch fabric adopting backpressure mode. However, the performance in the lossy mode is not acceptable for broadband communication systems where low levels of cell loss have to be guaranteed.

Several selection techniques, proposed in the literature and listed in [5], are used to resolve the routing decision in ideal nonblocking networks, such as the crossbar network. Recall that only one cell can be served by each output port in any switching cycle. Thus, these selection techniques are used to resolve the situation when more than one cell requests the same output destination. All these selection techniques

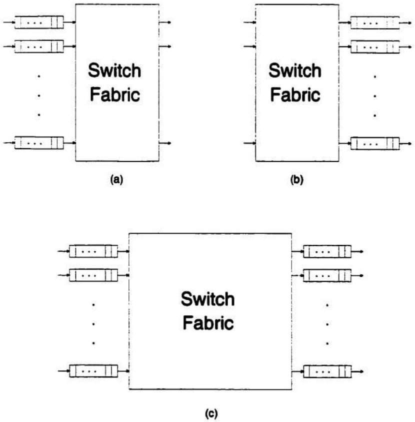


Figure 4.1: Input-output buffer strategies (a) pure-input (b) pure-output (c) input-output buffering

agree on that under uniform random traffic a pure-input buffered ideal nonblocking network has a maximum throughput (TP_{max}) of $2 - \sqrt{2} = 0.586$ as $N \to \infty$ and a FIFO input buffer size $B_{in} \to \infty$. This is due to HOL blocking problem encountered at the input buffers as a result of the backpressure mode. Clearly, using pure-input buffering strategy is not an optimal solution to improve the performance of switch fabrics.

In pure-output buffering strategy, buffers only exist at the output ports. In an ideal nonblocking fabric, the pure-output buffering strategy outperforms the pure-input buffering strategy provided that both buffers are of infinite size and $N \to \infty$. TP_{max} for the latter case is 100% compared to 58.6% for the former. The average cell delay in both pure-input (D_{in}) and pure-output buffering strategies (D_{out}) are given by [5]:

$$D_{in} = \frac{(2-p)(1-p)}{(2-\sqrt{2}-p)(2+\sqrt{2}-p)}; \qquad 0 \le p < 2-\sqrt{2}, \qquad (4.2)$$

$$D_{out} = \frac{p}{2(1-p)}; \qquad 0 \le p < 1, \qquad (4.3)$$

where p represents the applied load. Figure 4.2 depicts the delay performance of both strategies.

The superiority of the pure-output buffering strategy described above is not enjoyed under all conditions, especially when the buffer sizes are finite. Pure-output buffering does not improve the performance of some switching architectures at all. For example, those architectures where only one cell can be accepted by each output port in any switching cycle will not benefit from output buffering if the departure rate (sometimes called consumption rate) is 1 from the OPCs. The reason is that after each switching cycle no cell will be left to be buffered in the OPCs.

We conclude from the above discussion that pure-input and pure-output buffering

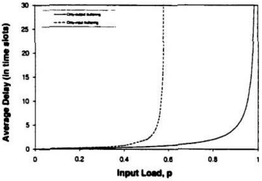


Figure 4.2: Behavior of D_{in} and D_{out}

strategies are not the best solutions. Instead, a hybrid of both strategies would lead to a better compromise. Several studies reported in [5] have supported this fact. With input-output buffering strategy we gain the advantages of both strategies. We gain the capability of the input buffers to combat internal and output blocking. We also gain the reduction of the HOL blocking enjoyed by the output buffering strategy. An ideal network does not need input buffering because the network does not suffer from any internal or output blocking.

4.2.2 Internal Buffering

In the literature [66], four internal buffering strategies have been reported. These strategies are depicted in Figure 4.3 for a 2 × 2 SE. Internal buffering is mainly used in single path MINs, because using internal buffering in multipath MINs results in an out-of-sequence problem. Several efforts have been reported [67] to study the performance of banyan networks with internal buffering and under different loads of uniform and non-uniform traffic. If we exclude the input buffering strategy, it is found that under uniform random traffic the banyan network with internal buffering

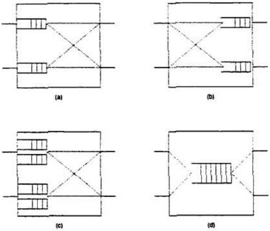


Figure 4.3: Different internal buffering styles (a) input (b) output (c) crosspoint (d) shared buffering

can attain TP_{max} of 100% when the buffering budget → ∞. In case of pure-input buffering, a TP_{max} of 75% is achieved under the same assumption of infinite buffering budget. Also all the studies have agreed that the shared buffering strategy has the best performance amongst all others if the buffer sizes are finite. However, shared buffering has the highest hardware complexity. Thus, it has been argued in [68] that crosspoint buffering strategy is the best choice if both complexity and performance are considered.

We mentioned in the previous section that real fabrics have limited buffer resources. As we approach these real limits we discover that internal buffering is not as attractive alternative as it appears with the assumption of infinite buffer resources. For example, in [6], the performance of a 64×64 banyan network with internal buffering was studied. The network was built from 4×4 SEs with each having a shared buffer of size 60 cells. The performance was studied using five different buffer manag-

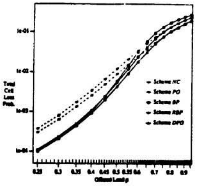


Figure 4.4: Performance of 64 × 64 Banyan with internal shared buffering strategy [6].

ing schemes under uniform and non-uniform traffic loads. The schemes are *no controls* (NC), *pushout* (PO), *backpressure* (BP), *restricted backpressure* (RBP), and *delayed pushout* (DPO). The simulation results showed that the best and the worst cell loss ratio levels are obtained when using DPO and NC schemes respectively. Even with this considerable amount of internal buffering, which is 2880 cells in the whole fabric, the obtained levels of cell loss ratio are very high as depicted in Figure 4.4 under URT. For example, when adopting the best internal buffering managing scheme, which is the DPO, the best cell loss ratio level of 10^{-4} can be only achieved under a load of 25%. Obviously, this performance is not acceptable for broadband applications. In general, internal buffering is not desirable for the following reasons:

- it results in out-of-sequence problem in multipath networks as we discussed earlier.
- it complicates the internal structures of the SEs. This is reflected on the design aspects, fault diagnosis, testing, area, speed, power planning ...etc.

- the complexity/performance ratio is high if compared to other solutions used to improve the performance of MINs; such as pipelining, replication, dilation.

In our analysis, we compare the performance of the BG network with both the crossbar and the ideal nonblocking networks. We have chosen the ideal nonblocking network because it has the best performance that can ever be achieved. We also have chosen the crossbar network because it has good internal nonblocking characteristics. The reference model used for the three fabrics is the same as the model shown in Figure 4.1.c. We decided to use only the input-output buffering strategy. That is, no internal buffering is adopted in the three fabrics due to the reasons discussed above.

We briefly provide an overview about the accuracy of the obtained simulation results in this dissertation. Let H be a random variable representing the outcome of one of our simulation experiments. We assume that H follows a normal distribution with both unknown mean μ and variance σ^2. A $100(1-\alpha)$ percent two-sided confidence interval on μ is given by [69]:

$$\bar{H} - t_{\alpha/2,m-1}S/\sqrt{m} \leq \mu \leq \bar{H} + t_{\alpha/2,m-1}S/\sqrt{m} \qquad (4.4)$$

where m is the number of simulation trials at each point, $\bar{H}$ is the average of the m simulation trials, S is the standard deviation of the m simulation trials, and $t_{\alpha/2,n-1}$ is the t-distribution with $n-1$ degrees of freedom. In all our simulation trials we set $\alpha = 0.05$ and $m = 20$.

4.3 Uniform Random Traffic

URT is perhaps the most common traffic used to study the performance of switch fabrics for the following reasons. Firstly, the overhead needed to generate the URT is small compared to other traffic models. Secondly, analytical modelling is feasible for almost every switching architecture. Thirdly, URT provides more realistic loads than the previously famous permutation traffic. In the literature, almost all the proposed switching architectures are mainly studied under URT loads. The URT assumes the following: 1) the traffic is arriving at the input ports with the same probability which is equal to the average applied load to the network ρ, and 2) the arriving cells at each of the input ports selects their destinations randomly with equal probability.

4.3.1 Analytical Modelling

Analytical modelling is the vehicle used to validate simulation results. In the literature, many efforts have been exerted to model the switching architectures under uniform and non-uniform traffic loads. The analytical model we introduce [70] here can be used to describe the performance of the BG network when implemented in either the pipelined fashion or parallel backpressure fashion discussed at the end of Section 2.4.2.2. Before we discuss our model in detail, we introduce the following notation:

K = Number of parallel planes.

t_r = Reservation cycle.

T = Switching cycle.

$p_i(t)$ = Probability that a buffer has i cells at switching cycle t.

$d(t)$ = Throughput after every t_r (t is in steps of t_r).

$TP(\tau)$ = Throughput after every T (τ is in steps of T).

Notice that:

$$p_0(t) = 1 - \sum_{i=1}^{B_{in}} p_i(t). \tag{4.5}$$

In our model we assume the following:

- The probability of cell arrival is the same for all input links.

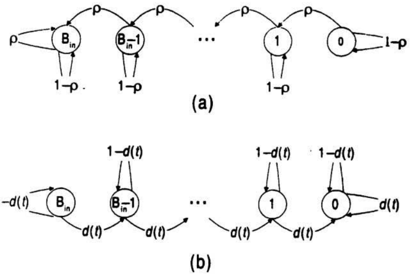


Figure 4.5: Discrete Markov chain of the input buffer status (a) at the beginning of every T, and (b) after every t_r.

- Arriving cells will request outputs with equal probability.
- Cells arrive only at the beginning of each switching cycle, that is, no fresh cells arrive in each reservation cycle.
- The cells depart from the output buffer at a rate of $4 \times K$ cells every switching cycle.

The first two assumptions represent URT and imply that any SE located in any stage is indistinguishable from the other SEs belonging to the same stage. As a result, each stage can be characterized by any of its SEs. The third assumption enables splitting the discrete Markov chain, which is used to describe the input buffer status, into two as shown in Figure 4.5. The fourth assumption is basically equivalent to assuming an infinite output buffer. Figure 4.5.a shows the Markov chain for the input buffer status at the beginning of every switching cycle, whereas Figure 4.5.b. shows the Markov chain for the input buffer status after every reservation cycle. From Figure 4.5.a we can write:

$$
\begin{aligned}
p_i(\tau+1) &= (1-\rho)p_i(\tau); & i=0, && (4.6)\\
&= \rho p_{i-1}(\tau) + (1-\rho)p_i(\tau); & 0<i<B_{in}, && (4.7)\\
&= p_{B_{in}}(\tau) + \rho p_{B_{in}-1}(\tau); & i=B_{in}. && (4.8)
\end{aligned}
$$

And from Figure 4.5.b we can write:

$$
\begin{aligned}
p_i(t+1) &= p_0(t) + d(t)p_1(t); & i=0, && (4.9)\\
&= d(t)p_{i+1}(t) + [1-d(t)]\,p_i(t); & 0<i<B_{in}, && (4.10)\\
&= [1-d(t)]\,p_{B_{in}}(t); & i=B_{in}, && (4.11)
\end{aligned}
$$

where

$$d(t) = \frac{X(t)}{1-p_0(t)} \qquad (4.12)$$

and $X(t)$ is the probability of propagating a cell from an input port of the BG network to the required output under URT. This is explained in Appendix C. The resulting throughput of the network is given by:

$$TP(\tau) = \frac{\sum_{i=1}^{K} X(i \times t_r)}{\rho}. \qquad (4.13)$$

The buffer status is initialized to represent an empty buffer as follows:

$$p_0(0) = 1, \qquad p_1(0) = p_2(0) = \ldots = p_{B_{in}}(0) = 0. \qquad (4.14)$$

Table 4.1 introduces both simulation and analytical results of the TP_{max} with $B_{in} = 0$. Notice that a unity TP_{max} for the simulation results does not necessarily mean zero cell loss; it means no cell was lost during our simulation trials. The number of cells, which were applied in any of our simulation experiments was at least 10^7. That is

	$K = 1$		$K = 2$	
N	Sim.	Ana.	Sim.	Ana.
2	1	1	1	1
4	1	1	1	1
8	0.992602	0.993351	1	1
16	0.98462	0.986201	1	1
32	0.976348	0.9791746	1	1
64	0.967142	0.9723303	1	1
128	0.958486	0.965666	0.999969	1
256	0.949810	0.9591744	0.999954	0.9999998
512	0.941769	0.9528478	0.999947	0.9999997
1024	0.934461	0.9466793	0.999924	0.9999995

Table 4.1: Analytical and simulation results for the BG network with zero input buffering

a cell loss level of 10^{-7} is assured in case of unity TP_{max}. We observed that as we increase the capacity of the input buffer to two cells for a network of two data planes, we obtain a unity TP_{max} for both analytical and simulation results. This describes the efficiency of the BG network. We also found out that the size of the input buffer marginally affects the performance of the network composed of only one data plane as shown in Table 4.2. This agrees with the fact, we will discover later, that input queuing has less impact on the performance of the network than output queuing.

4.3.2 Finite Output Buffer

In this section, we investigate the performance of the BG network compared with the crossbar and the ideal nonblocking networks. BG-K is a pipelined BG network composed of K data planes. Similarly, Crossbar-K is a pipelined Crossbar network composed of K data planes. These results are obtained for a buffer budget of 5000 cells per input and output queue. Except for the crossbar network, each input queue length is set to 1000 cells and output queue is set to 4000 cells. Since the consumption rate is assumed to be 1 in all our simulation experiments, no output buffering is

	$B_{in} = 1$		$B_{in} = 2$		$B_{in} = 5$		$B_{in} = 150$	
N	Sim.	Ana.	Sim.	Ana.	Sim.	Ana.	Sim.	Ana.
2	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1
8	0.993113	0.993351	0.993201	0.993351	0.993298	0.993351	0.993326	0.993351
16	0.985171	0.986201	0.985971	0.986201	0.986031	0.986201	0.986087	0.986201
32	0.976681	0.979175	0.976801	0.979175	0.976903	0.979175	0.976947	0.979175
64	0.967250	0.972330	0.967301	0.972330	0.967393	0.972330	0.967427	0.972330
128	0.957012	0.965666	0.958170	0.965666	0.959601	0.965666	0.959646	0.965666
256	0.950440	0.959174	0.950701	0.959174	0.950965	0.959174	0.951007	0.959174
512	0.942348	0.952848	0.942553	0.952848	0.942620	0.952848	0.942722	0.952848
1024	0.934751	0.946679	0.934858	0.946679	0.935001	0.946679	0.935060	0.946679

Table 4.2: Effect of input buffer size on the TP_{max} of a single data plane BG network

Load	Ideal	BG-1	Crossbar-1	Crossbar-2
10	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$
20	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$
30	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$
40	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$
50	$< 10^{-7}$	$< 10^{-7}$	$0.004 \pm 1.81 \times 10^{-5}$	$< 10^{-7}$
60	$< 10^{-7}$	$< 10^{-7}$	$0.06376 \pm 3.07 \times 10^{-5}$	$< 10^{-7}$
70	$< 10^{-7}$	$< 10^{-7}$	$0.14117 \pm 4.39 \times 10^{-5}$	$< 10^{-7}$
80	$< 10^{-7}$	$< 10^{-7}$	$0.21374 \pm 2.95 \times 10^{-5}$	$< 10^{-7}$
90	$< 10^{-7}$	$< 10^{-7}$	$0.27722 \pm 2.81 \times 10^{-5}$	$< 10^{-7}$

Table 4.3: Cell loss under URT for different network types of $N = 256$.

required for the crossbar-1 network. Accordingly, all the buffer budget is applied only to the input queues of the crossbar-1 network. For the crossbar-2 network each input queue is set to 2500 cells and each output queue is set to 2500 cells. For the other crossbar-K networks, where $K \geq 3$ the input queues are set at 1000 cells and the output queues are set at 4000 cells. Also the network size N is set to 256.

Table 4.3 provides the performance of the cell loss for the networks under consideration. Both Figures 4.6 and 4.7 show the performance of the cell delay. The buffering requirements are also depicted in Table 4.4. All the above results suggest that a single data plane BG network and the crossbar-2 network are sufficient to satisfy the requirements of the URT. It is also noticeable that the performance of the BG-1 network is the closest to the ideal network. HOL blocking at the input buffers is the main reason that the input buffering requirements of the crossbar configurations are larger. HOL blocking is a consequence of the inefficient cell relaying behavior of the crossbar network. This inefficient behavior makes the crossbar architectures rely more on the input buffers rather than the output buffers. As we discussed in Section 4.2.1, HOL blocking is more profound in input buffered networks. Channel grouping, virtual output queues, and windowing are well-known mechanisms used to ease HOL blocking in input buffered architectures as described in [71]. However, these mechanisms lead to the out-of-sequence problem. Besides, their hardware implementation is intensive.

We conjecture the sufficiency of both the BG and the crossbar-2 networks for the requirements of the URT is due to the nature of the URT itself. Because the output ports are selected randomly with equal probabilities, Bernoulli distribution can be used to represent the probability ($P_{Req=i}$) that an output port is requested by i input ports in any switching cycle as:

$$P_{Req=i} = \frac{\binom{N}{i}(N-1)^{N-i}}{N^N}; \qquad 0 \le i \le N \qquad (4.15)$$

Figure 4.8 is based on Equation 4.15. Figure 4.8 tells that almost 99% of the input port requests to a single output port are confined to the range from 0 up to 4. Since, in any switching cycle, the crossbar-2 can only deliver 2 cells at most to any single output port, its input buffering requirements are larger than BG-1 which can deliver up to 4 cells to a single output port. This also explains the better delay performance of the BG-1 network over the crossbar-2 network.

4.4 Bursty Traffic

Bursty traffic is a traffic type in which the inputs of the switch fabric receive sudden bursts of packets [72]. The On-Off model is the least complex and the most widely used to model bursty sources in simulation. The On-Off model can describe most of the existing sources with a reasonable accuracy [73]. The model assumes that the source generates cells in a bursty manner: one active period (On period) followed by an idle one (Off period). The lengths of On and Off periods are independently

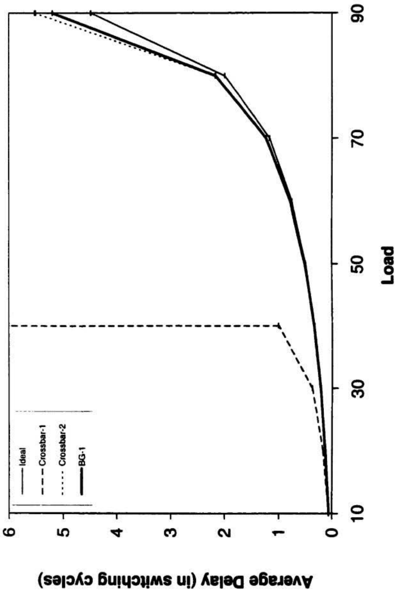


Figure 4.6: Average cell delay under URT for different network types of $N = 256$.

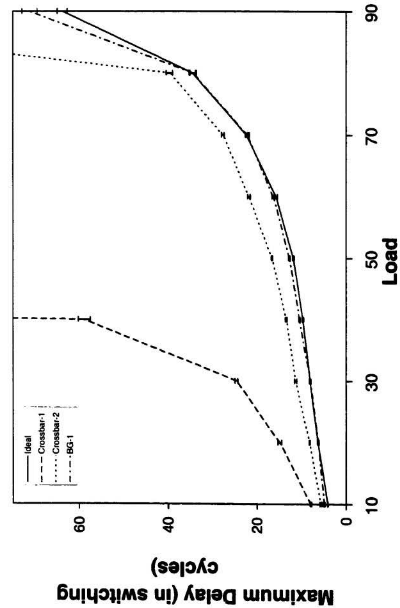


Figure 4.7: Maximum cell delay under URT for different network types of $N = 256$.

Load	Ideal		BG-1		Crossbar-1		Crossbar-2	
	IN	OUT	IN	OUT	IN	OUT	IN	OUT
10	0	4.1 ± 0.06	1	4.6 ± 0.10	4	0	2	3.4 ± 0.10
20	0	6.3 ± 0.93	1.8 ± 0.08	6.1 ± 0.06	6.6 ± 0.14	0	2.9 ± 0.06	5.2 ± 0.08
30	0	8.1 ± 0.17	2	7.6 ± 0.10	11.5 ± 0.16	0	3.5 ± 0.10	6.8 ± 0.08
40	0	9.7 ± 0.16	2.6 ± 0.10	10 ± 0.13	27.1 ± 0.53	0	4.7 ± 0.09	9.1 ± 0.11
50	0	11.9 ± 0.17	3.1 ± 0.06	12.7 ± 0.16	5000 +	0	6.1 ± 0.11	11.9 ± 0.19
60	0	15.8 ± 0.27	4.1 ± 0.06	16.3 ± 0.26	5000 +	0	8.2 ± 0.08	15.6 ± 0.33
70	0	22.5 ± 0.28	5 ± 0.09	22.1 ± 0.43	5000 +	0	12.1 ± 0.17	20.8 ± 0.41
80	0	34.5 ± 0.82	6.9 ± 0.11	33.8 ± 0.58	5000 +	0	20.2 ± 0.24	32.5 ± 0.46
90	0	63.8 ± 1.13	15.6 ± 0.31	68.2 ± 1.80	5000 +	0	112.2 ± 2.43	66.6 ± 1.71

Table 4.4: Input-output buffering requirements under URT for different network types of $N = 256$.

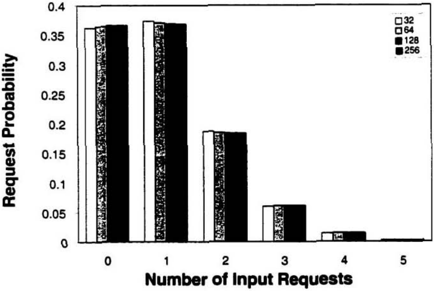


Figure 4.8: Probability density for number of input requests for a single output port.

evaluated from two geometric distributions given by [74]:

$$L = 1 + \lceil \frac{\ln(1 - R)}{\ln(1 - p)} - 1 \rceil, \tag{4.16}$$

where L is the period length in cells, $0 \leq R \leq 1$ is the random number generated, and $0 < p \leq 1$ is inverse of the average period length in cells. The cells arriving at each input line in a burst are destined to the same output line. This output is selected randomly. URT can be considered as a special case of the bursty traffic with $L = 1$.

Table 4.5 depicts the cell loss ratio for single plane configurations under loads of bursty traffic of various burst lengths. The results in Table 4.5 imply that a single data plane BG network can attain very low levels of cell loss under heavy bursty traffic loads of average burst lengths $L \leq 10$. Moreover, the BG-1 network has good cell loss levels with $L > 10$.

Figures 4.9, 4.10, 4.11, and 4.12 represent performance of the average cell delay, maximum cell delay, input buffer requirements, and output buffer requirements, respectively, for single data plane configurations of the three networks under test. The results suggest that the performance of the BG-1 configuration outperforms the performance of the crossbar-1 configuration. For example, the average cell delay of the BG-1 configuration almost coincides with the average cell delay of the ideal network as depicted in Figure 4.9.a. However, for $L \geq 15$ and load greater than 80% the average cell delay of the BG-1 network grows faster than the average cell delay of the ideal network. The reason is that as the burst length increases, longer bursts are created at the IPCs. These bursts concentrates for longer periods on their destined OPCs raising the probability of output blocking. A similar behavior can be observed for the maximum cell delay in Figure 4.10.

As shown in Figure 4.11, the input buffer of the crossbar-1 overflows at a load less than 40%, which is the load after which the input buffer overflows for the crossbar-1

		L=5	
Load	Ideal	BG-1	crossbar-1
≤ 30	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$
40	$< 10^{-7}$	$< 10^{-7}$	$0.00063 \pm 2.12 \times 10^{-5}$
50	$< 10^{-7}$	$< 10^{-7}$	$0.0529 \pm 8.11 \times 10^{-5}$
60	$< 10^{-7}$	$< 10^{-7}$	$0.17024 \pm 8.7 \times 10^{-5}$
70	$< 10^{-7}$	$< 10^{-7}$	$0.24295 \pm 5.24 \times 10^{-5}$
80	$< 10^{-7}$	$< 10^{-7}$	$0.32588 \pm 7.48 \times 10^{-5}$
90	$< 10^{-7}$	$< 10^{-7}$	$0.3685 \pm 8.51 \times 10^{-5}$
		L=10	
Load	Ideal	BG-1	crossbar-1
≤ 30	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$
40	$< 10^{-7}$	$< 10^{-7}$	$0.00202 \pm 6.57 \times 10^{-5}$
50	$< 10^{-7}$	$< 10^{-7}$	0.06017 ± 0.00013
60	$< 10^{-7}$	$< 10^{-7}$	0.18038 ± 0.00016
70	$< 10^{-7}$	$< 10^{-7}$	0.25290 ± 0.00014
80	$< 10^{-7}$	$< 10^{-7}$	$0.33545 \pm 7.2 \times 10^{-5}$
90	$< 10^{-7}$	$< 10^{-7}$	0.37545 ± 0.00012
		L=15	
Load	Ideal	BG-1	crossbar-1
≤ 30	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$
40	$< 10^{-7}$	$< 10^{-7}$	$0.00027 \pm 2.19 \times 10^{-5}$
50	$< 10^{-7}$	$< 10^{-7}$	0.06279 ± 0.00015
60	$< 10^{-7}$	$< 10^{-7}$	0.1836 ± 0.00015
70	$< 10^{-7}$	$< 10^{-7}$	0.25633 ± 0.00015
80	$< 10^{-7}$	$< 10^{-7}$	$0.33836 \pm 7.89 \times 10^{-5}$
90	$< 10^{-7}$	$9.90096 \times 10^{-6} \pm 1.68872 \times 10^{-6}$	0.39495 ± 0.00011
		L=20	
Load	Ideal	BG-1	crossbar-1
≤ 30	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$
40	$< 10^{-7}$	$< 10^{-7}$	$0.00092 \pm 6.01 \times 10^{-5}$
50	$< 10^{-7}$	$< 10^{-7}$	0.06410 ± 0.00041
60	$< 10^{-7}$	$< 10^{-7}$	0.16577 ± 0.00032
70	$< 10^{-7}$	$< 10^{-7}$	0.25745 ± 0.00015
80	$< 10^{-7}$	$< 10^{-7}$	0.33971 ± 0.00021
90	$< 10^{-7}$	$7.71046 \times 10^{-4} \pm 9.13291 \times 10^{-6}$	0.41006 ± 0.00013

Table 4.5: Cell loss ratio under bursty loads of various burst lengths.

under URT. We can also deduce that the input buffer of the crossbar-1 overflows at lower loads as L increases. Similar to the results we obtained under URT loads, the ideal network has no input buffer requirements for the burst lengths we use. As expected, the buffering requirements of the BG-1 configuration diverges from the ideal network for $L \geq 15$ and load greater 80%. As depicted in Figures 4.11.c and 4.11.d, the input buffer overflows for the BG-1 under the aforementioned loads. This explains the higher cell loss ratio ($> 10^{-7}$) for the BG-1 configuration in Table 4.5. Figure 4.12 implies that, and similar to our observation for the performance under URT, the crossbar-1 configuration does not need any output buffering.

To have a deep understanding of the crossbar and the BG networks we compare the performance of their configurations, which have more planes ($K > 1$), with the ideal network performance. Two immediate advantages are obtained by using configurations of multiple planes. Firstly, the performance is greatly enhanced for low performance architectures, such as the crossbar-1 configuration. Secondly, the system availability is improved because failure in any of the planes is compensated by the other functional ones, although the performance degrades as one or more planes are out of service. Figure 4.13 depicts the number of data planes needed to reach 10^{-7} cell loss ratio, for each of the three networks, under bursty traffic loads up to 90% of different average burst length. Our criteria to compare the performance of the three architectures under test will be based on the number of planes shown in Figure 4.13 and listed in Table 4.6.

Figures 4.14, 4.15, 4.16, and 4.17 depict the simulation results of the average cell delay, maximum cell delay, input buffer requirements, and output buffer requirements, respectively, for the configurations listed in Table 4.6. The results suggest that the delay performance of the proposed configurations are very similar as depicted in Figures 4.14 and 4.15. However, the results also imply that running the switching

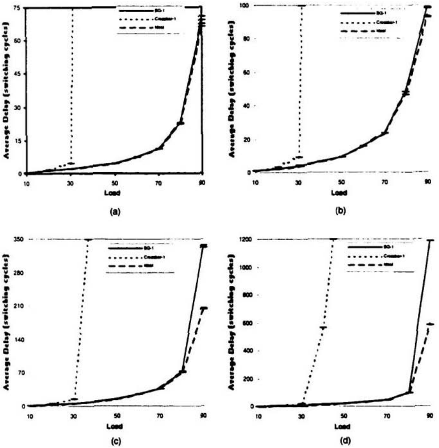


Figure 4.9: Average cell delay for different average burst length (a) $L = 5$, (b) $L = 10$, (c) $L = 15$, (d) $L = 20$.

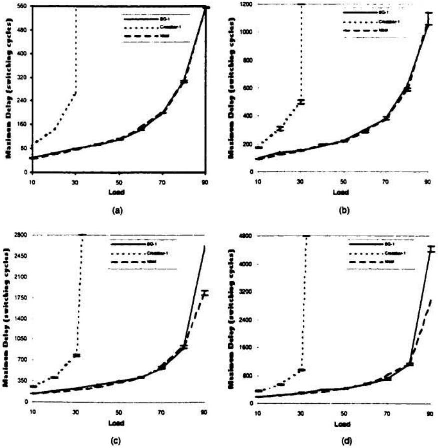


Figure 4.10: Maximum cell delay for different average burst length (a) $L = 5$, (b) $L = 10$, (c) $L = 15$, (d) $L = 20$.

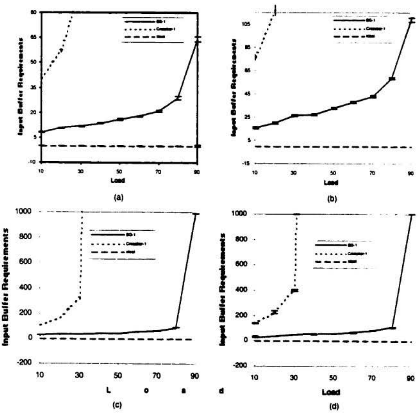


Figure 4.11: Input buffer requirements for different average burst length (a) $L = 5$, (b) $L = 10$, (c) $L = 15$, (d) $L = 20$.

L	BG	Crossbar	Ideal
5	1	3	1
10	1	3	1
15	2	3	1
20	2	3	1

Table 4.6: Number of planes followed to compare the multiple plane configurations of the architectures under test.

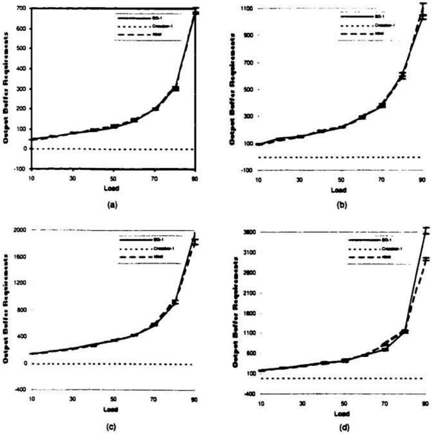


Figure 4.12: Output buffer requirements for different average burst length (a) $L = 5$, (b) $L = 10$, (c) $L = 15$, (d) $L = 20$.

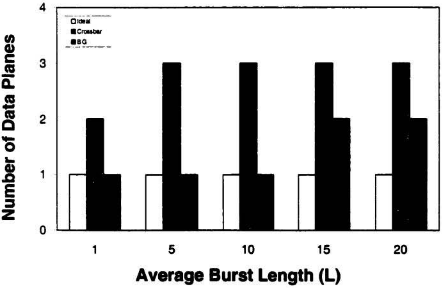


Figure 4.13: Required number of planes to attain a 10^{-7} cell loss ratio with different average burst lengths.

networks under heavy loads is not recommended due to the high delays experienced by the traffic waiting in the input and output queues of the network. This delay can exceed 3600 switching cycles under 90% bursty load with $L = 20$ as depicted in Figure 4.15.d. This also applies to the ideal network, where a maximum cell delay of almost 3000 switching cycles can be experienced as shown in Figure 4.15.d.

The output buffering requirements are also very similar for the three networks as depicted in Figure 4.17, except under loads greater than 80% and $L \geq 5$. However. the input buffering requirements differ as depicted in Figure 4.16. As expected, the ideal network has the lowest input buffering requirements followed by the BG network. The reason that the crossbar-3 configuration has the highest input buffering requirements is because it is less efficient than the Ideal and the BG configurations in relaying the traffic from the IPCs to the OPCs. The drop in the input buffer requirements for the BG configurations as shown in Figures 4.16.b and 4.16.c is because the BG configuration in Figure 4.16.b consists of one plane, whereas it consists of two planes in Figure 4.16.c. With the input buffering requirements of the crossbar-3 configuration are the highest, one should expect the that output buffering requirements of the crossbar configuration should be lowest. But the fact is, the routing algorithm of the crossbar architecture does not treat all the OPCs with equal priority. Hence, those OPCs that are favored need more buffering capacity giving rise to the output buffering requirements of the crossbar configurations. The impact of that effect lessens as the number of planes increases. A fair routing algorithm for the crossbar network is expensive in terms of hardware complexity [19]. Obviously, this is an advantage of the BG architecture over the crossbar architecture.

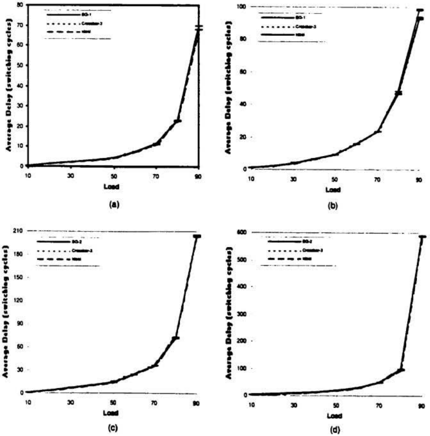


Figure 4.14: Average cell delay for different average burst length (a) $L = 5$, (b) $L = 10$, (c) $L = 15$, (d) $L = 20$.

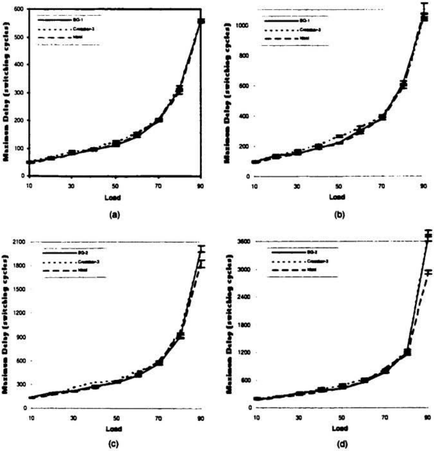


Figure 4.15: Maximum cell delay for different average burst length (a) $L = 5$, (b) $L = 10$, (c) $L = 15$, (d) $L = 20$.

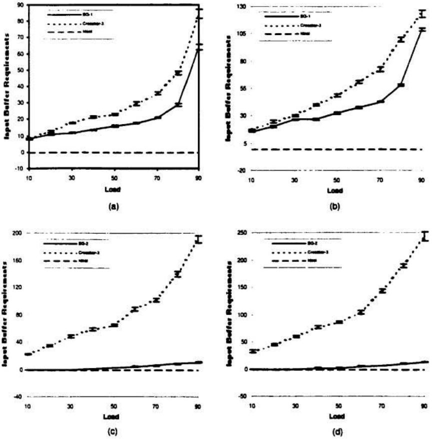
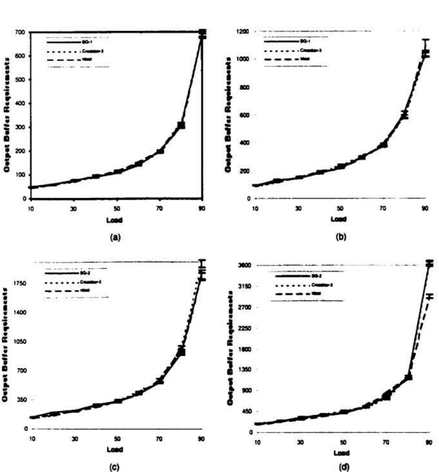


Figure 4.16: Input buffer requirements for different average burst length (a) $L = 5$, (b) $L = 10$, (c) $L = 15$, (d) $L = 20$.



(a)

(b)

(c)

(d)

Figure 4.17: Output buffer requirements for different average burst length (a) $L = 5$, (b) $L = 10$, (c) $L = 15$, (d) $L = 20$.

4.5 Non-Uniform Traffic

In all the traffic loads discussed previously, we assumed that the OPCs are equally (randomly) selected by the arriving traffic at the IPCs. However, in real broadband communication networks it is expected that the arriving traffic will select the OPCs in a non-uniform way. In this section, we use a modified form of the model presented in [7] for non-uniform traffic generation. The model in [7] assumes that the OPCs mapping for each incoming cell is determined by the binomial distribution, where for $1 \leq i \leq N-2$:

$$\begin{aligned} a_{i,j} &= \Pr[\text{a cell arriving at the } i\text{-th input is destined to the } j\text{-th output}], \\ &= \binom{N-1}{j} r_i^j (1-r_i)^{N-j-1}; \qquad 0 \leq j \leq N-1 \end{aligned} \tag{4.17}$$

and r_i is the probability associated with input i. For the binomial distribution, the maximum probability occurs for $j = \lfloor N r_i \rfloor$. If OPC_{N-i-1} is chosen to receive the highest percentage of the traffic arriving at input i, then we get:

$$r_i = 1 - \frac{i}{N-1} \tag{4.18}$$

For IPC_0 and IPC_{N-1}, the address of OPC_j, where $0 \leq j \leq N-1$, is given by the normalized Possion-like distribution with rate r as follows:

$$a_{0,j} = \frac{\frac{r^{N-j-1}}{(N-j-1)!}}{\sum_{\forall j} \frac{r^{N-j-1}}{(N-j-1)!}}, \qquad a_{N-1,j} = \frac{\frac{r^j}{j!}}{\sum_{\forall j} \frac{r^j}{j!}}. \tag{4.19}$$

The advantage of the above model is that it gives a substantial number of hot spots (instead of only one or two), which is more realistic as the number of the switch ports is relatively high. However, the above model deals differently with the IPCs as IPC_0 and IPC_{N-1} follow a different distribution. Also, the value of r_i for the other IPCs is different which results in different distribution for each IPC as depicted in Figure 4.18.

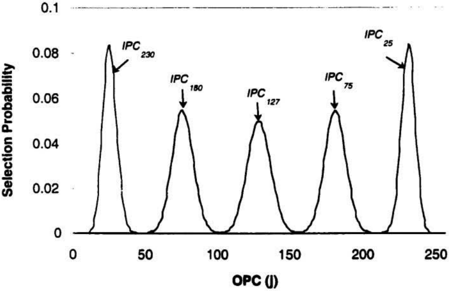


Figure 4.18: OPCs selection probability for different IPCs according to the model in [7].

Load	BG	Crossbar	Ideal
10	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$
10	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$
30	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$
40	$< 10^{-7}$	$< 10^{-7}$	$< 10^{-7}$
50	$< 10^{-7}$	$6.44172 \times 10^{-5} \pm 4.415 \times 10^{-6}$	$< 10^{-7}$
60	$< 10^{-7}$	0.04941 ± 0.000166	$< 10^{-7}$
70	$< 10^{-7}$	$0.123943 \pm 8.22 \times 10^{-5}$	$< 10^{-7}$
80	$< 10^{-7}$	0.196649 ± 0.000104	$< 10^{-7}$
90	$< 10^{-7}$	$0.260677 \pm 8.59 \times 10^{-5}$	$< 10^{-7}$

Table 4.7: Cell loss ratio for single plane architectures under non-uniform traffic load.

In our modified model, we assume that all the IPCs will select the OPCs with the same binomial distribution with a fixed value of r_i. In the beginning of each simulation experiment, the value of r_i is determined and each IPC is randomly associated with an OPC, not to be associated with any other IPC. For each IPC, the associated OPC is located at the center of the binomial distribution. That way we solve the problems we discussed above and add another advantage of randomizing the OPC selection in the beginning of the simulation experiment. We first study the performance if the arriving traffic has an average burst length $L = 1$ and then later we study the performance under different burst lengths ($L > 1$).

4.5.1 $L = 1$

Table 4.7 lists the cell loss ratio for single plane configurations of the architectures under test. As shown, crossbar-1 configuration can not satisfy the 10^{-7} cell loss ratio for loads above 40%. While, the BG-1 configuration can satisfy the 10^{-7} cell loss ratio for loads up to 90%. Figure 4.19 shows the behavior of the performance parameters of the various single plane configurations and crossbar-2 configuration under different loads of non-uniform traffic with $L = 1$. Up to 90% load, the performance of the BG-

1 configuration is almost coinciding with the performance of the ideal architecture. Again, the performance crossbar-1 configuration lags far behind both the BG-1 and the ideal configurations. Although, the crossbar-2 could fulfill the 10^{-7} cell loss ratio, its performance could not also match the performance of the BG-1 configuration as shown in Figure 4.19. This result is similar to what we found out under URT loads.

One interesting observation can be found by comparing the performance of similar configurations when operated under the URT and the non-uniform load. In general, we can observe a general enhancement in the performance under non-uniform loads for all configurations. However, this enhancement is more pronounced in the crossbar configurations. For example, in Figure 4.19.c the input buffer requirements for the crossbar-2 configuration under 90% non-uniform load is almost 37. While in Table 4.4 under 90% URT load the crossbar-2 configuration has input buffer requirements of 112. Under non-uniform loads, the chances of multiple IPCs destining to one single OPC are lessened because each source is now targeting certain band of the OPCs. Accordingly, the non-uniform traffic of $L = 1$, to some extent, resembles the permutation traffic. We know from the discussions in Section 2.4.1 that the crossbar architecture has a perfect performance under permutation traffic.

4.5.2 $L > 1$

The results we obtained under non-uniform burst loads for single plane configurations support the same argument we stated under non-uniform loads with $L = 1$ that there is a slight enhancement in the performance under non-uniform traffic loads. This could be explained by the same reason we have given above. However, we observed an interesting phenomenon for the BG-1 configuration. Although the delay (average and maximum) performance is slightly better under non-uniform bursty loads than under uniform bursty loads, the input buffer requirements are found to be slightly

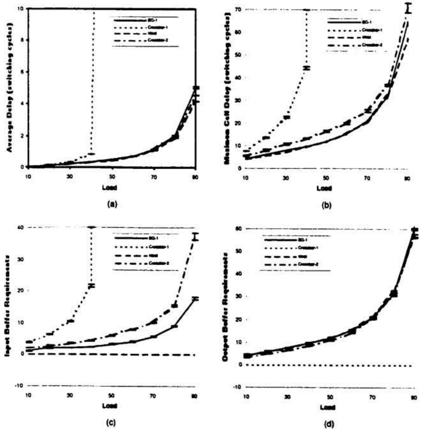


Figure 4.19: Performance parameters of different configurations under different loads of non-uniform traffic.

Input Buffer Requirements				
Load	$L = 5$		$L = 10$	
	Nonuni	Uni	Nonuni	Uni
10	9 ± 0.316	8.1 ± 0.308	15 ± 0.590	15.7 ± 0.644
20	10.8 ± 0.254	10.8 ± 0.313	21.2 ± 0.384	20.3 ± 0.670
30	11.3 ± 0.159	11.7 ± 0.130	23.3 ± 0.258	26.6 ± 0.774
40	13.2 ± 0.238	13.5 ± 0.245	26.3 ± 0.508	26.9 ± 0.600
50	15.7 ± 0.376	15.8 ± 0.313	30.7 ± 0.676	32.7 ± 0.651
60	19.3 ± 0.408	17.7 ± 0.316	37.8 ± 0.798	37.6 ± 0.785
70	23.4 ± 0.563	21 ± 0.446	46.7 ± 1.249	42.9 ± 0.621
80	40 ± 2.205	28.8 ± 0.849	84.7 ± 4.363	58.5 ± 0.724
90	81.3 ± 2.051	64.06 ± 1.610	142.1 ± 3.493	109 ± 1.617
Load	$L = 15$		$L = 20$	
	Nonuni	Uni	Nonuni	Uni
10	25.6 ± 1.302	23.8 ± 0.750	28.6 ± 0.934	32.2 ± 1.927
20	31.8 ± 0.761	30.9 ± 1.200	39.5 ± 0.990	38.2 ± 0.722
30	36.9 ± 1.052	35.9 ± 1.008	45.4 ± 1.423	47.2 ± 0.997
40	37.9 ± 1.028	40.9 ± 1.168	145.9 ± 57.989	53.8 ± 1.723
50	44.2 ± 1.005	43.1 ± 0.634	59.9 ± 1.377	55.9 ± 1.079
60	56.5 ± 1.712	53.1 ± 1.079	69.2 ± 1.307	67 ± 1.384
70	62.8 ± 1.353	64.6 ± 1.011	87.5 ± 2.988	82.8 ± 1.326
80	120.2 ± 6.955	87.1 ± 1.786	151.4 ± 4.483	108.7 ± 2.714
90	1000 ± 0.000	988.6 ± 4.895	1000 ± 0.000	1000 ± 0.000

Table 4.8: Input buffer requirements for BG-1 configuration under uniform and non-uniform bursty loads.

higher under non-uniform bursty loads than under uniform bursty loads. On the contrary, the output buffer requirements under non-uniform bursty loads are lower than under uniform bursty loads. Tables 4.8 and 4.9 depict that phenomenon. We attribute this phenomenon to the internal blocking characteristics of the BG architecture. As we mentioned above, non-uniform traffic resembles permutation traffic. That is, output blocking is less pronounced under non-uniform traffic. The lower output buffer requirements indicate that the longer input buffer queues are not due to overloaded output queues. The only other factor which can lead to the increased input buffer requirements is internal blocking. However, the overall performance of

Output Buffer Requirements				
Load	$L = 5$		$L = 10$	
	Nonuni	Uni	Nonuni	Uni
10	49.7 ± 1.497	49.1 ± 1.418	89.9 ± 2.166	93.6 ± 4.545
20	60.7 ± 1.055	60.5 ± 1.314	113.8 ± 2.834	134.2 ± 4.376
30	77.1 ± 2.236	77.6 ± 1.327	140.8 ± 3.610	148 ± 3.160
40	89.8 ± 1.451	92 ± 1.232	184.6 ± 5.825	184.3 ± 3.770
50	102.8 ± 1.515	109.1 ± 2.100	211.1 ± 5.046	219.7 ± 2.673
60	147.7 ± 4.442	139.4 ± 1.406	287.7 ± 4.968	297.8 ± 4.186
70	187.6 ± 2.344	198.8 ± 4.796	337.5 ± 2.938	373.3 ± 6.872
80	286.7 ± 5.465	305.7 ± 4.519	597.2 ± 16.250	611.4 ± 13.319
90	598.4 ± 18.714	684.2 ± 9.286	893.1 ± 17.820	1033.4 ± 15.492
Load	$L = 15$		$L = 20$	
	Nonuni	Uni	Nonuni	Uni
10	132.6 ± 2.486	141 ± 4.237	194.7 ± 7.999	180.4 ± 7.026
20	159.6 ± 2.731	176.2 ± 4.843	218.1 ± 6.172	233.4 ± 6.141
30	200.2 ± 2.435	214 ± 5.540	277.4 ± 6.836	293 ± 6.667
40	267.9 ± 9.372	279.5 ± 7.211	346.3 ± 223.230	370.2 ± 5.885
50	300.4 ± 6.622	338.3 ± 6.829	372.1 ± 6.808	402 ± 7.007
60	417.5 ± 8.401	410.7 ± 8.834	516.5 ± 8.245	551.6 ± 15.530
70	529.3 ± 12.017	563.9 ± 10.356	705.8 ± 21.962	699.1 ± 21.037
80	849.5 ± 16.313	903.1 ± 16.120	1093.6 ± 14.090	1121 ± 23.609
90	1630.3 ± 33.184	1944.1 ± 32.414	2988.8 ± 15.551	3634.9 ± 75.927

Table 4.9: Output buffer requirements for BG-1 configuration under uniform and non-uniform bursty loads.

the BG-1 configuration is better under non-uniform burst loads. This stems from the fact that the effect of internal blocking in the BG architecture is minimal. Previously, it has been shown [65] that the BG architecture has very low internal blocking characteristics.

4.6 Summary

The numerical results in this chapter have demonstrated the efficient performance of the BG network under various traffic loads. With finite buffering, a single data plane 256 × 256 BG network has been shown that it is sufficient to meet the requirements of the URT loads up to 90%. While, it has been discovered that a two plane crossbar performs less efficiently with higher buffer requirements under the same loading conditions. This is an indication that the routing capability of the BG network is more efficient than the crossbar based configurations. Additionally, a two plane 256 × 256 BG network has shown to be sufficient to meet the requirements of the bursty traffic of different average burst lengths. A three plane crossbar configuration is found to be less efficient with higher buffering requirements under uniform bursty traffic loads. Furthermore, a two plane BG configuration had lower buffering requirements than a two data plane crossbar configuration under non-uniform traffic loads. This is an interesting result because the non-uniform traffic model we used resembles the permutation traffic, which is the traffic type under which a single data plane crossbar network has an exact zero cell loss ratio. In general, we noticed that the performance of the BG network configurations is closer to the ideal nonblocking network. It is obvious from the above discussions that the BG architecture is a very strong candidate for broadband communication networks.

Chapter 5

Design of the Balanced Gamma Network

5.1 Introduction

In this chapter, we introduce the design of the BG network. We first describe the design methodology recommended by CMC for the design flow using deep submicron (DSM) technology. Because our design features built-in self-test (BIST), we review the BIST methods used in the design of VLSI systems, shedding light on the BIST mechanism we adopt in the BG network. Then, we illustrate the architectural design of the BG network in detail. The structure of each module in the design hierarchy is discussed, emphasizing the complexity and timing requirements. We also describe our approach for design verification of the whole system. Finally, we present the simulation results for each module in the system.

5.2 Design Flow, Functional Test and Verification

The design flow followed in this dissertation is based on the design flow recommended by CMC for DSM technologies [75]. As depicted in Figure 5.1, the first four steps of the design flow are carried out using Synopsys CAD tools and the rest are carried out using Cadence CAD tools. In the beginning, the architecture of the network is described in VHDL at the RTL level followed by simulation to verify the functionality. The next step is to synthesize the RTL description. It is recommended in the Synopsys documentation that large designs, such as the BG network, not be imported directly to the synthesis tool. Instead, a hierarchical bottom-up approach should be followed. This is because importing large designs leads to crashing the synthesis tool and in some cases may result in an unoptimized design. Accordingly, we divided the architecture of the BG network into modules. Each module is designed, synthesized, and tested separately and used as a building block in forming the final network design.

In general, dividing the design of any system has several merits. Firstly, the debugging process is simpler when building high level modules from low level ones. For example, sometimes the Synopsys synthesis CAD tool produces logical errors in the resulting gate level description of the design. It becomes intractable to detect and correct these logical errors in large designs. With a design composed of smaller submodules, one can more easily point at the submodules that are malfunctioning. Secondly, we can comfortably introduce testing features to the system under consideration. In fact, the process of building a testing mechanism may fail for large systems, because the larger the system the more the internal parts become inaccessible. The third reason is the reusability of the designed modules in future systems.

Finally, the design process was carried out to the fourth step, gate-level simulation, shown in Figure 5.1.

5.3 Design for Testability

To meet the quality and reliability requirements of today's complex communication networks, efficient testing methodologies are necessary at all levels of system design (system, board, IC, ...etc.). In practice, the quality of such systems is measured by

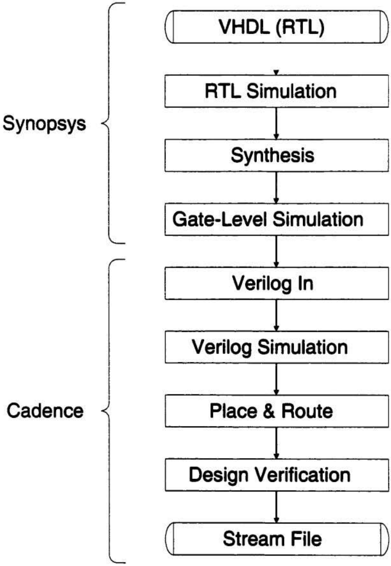


Figure 5.1: Design flow recommended by CMC.

the average time during which it functions correctly without any fault, and is termed mean time to failure (MTTF) as we will see in Chapter 6. High MTTF guarantees uninterrupted reliable network services to customers. This can be achieved by incorporating built-in test capabilities that monitor the system functionality periodically in the field against any failures, as well as reliable hardware components that are thoroughly tested for structural defects. In [8], a comprehensive and detailed treatment of digital systems testing can be found.

The conventional test methods in digital circuits are constantly challenged by increasing speed and circuit size which demand sophisticated automatic test equipment (ATE). This results in very high costs associated with test hardware, test generation, and test application time. Furthermore, at-speed testing (verification of system functionality at the rated speed) requires high-performance ATE, which results in multifold increase in their price. BIST offers a test methodology where the test functions are embedded into the circuit itself. The test functions in BIST are localized to the circuit, thereby facilitating at-speed test and substantial reduction in test application time. Furthermore, BIST provides easy access to the embedded components and interconnections of the system without any special test requirements. For the above reasons we decided to adopt BIST strategy in the design of BG network for testing.

In [76], an overview of the digital BIST techniques that are particularly applicable to telecommunication systems is presented. This article discussed the BIST techniques used at different system levels. In the next section we discuss BIST methods in more detail.

5.3.1 BIST Methods

BIST methods can be divided into two groups as depicted in Figure 5.2. In on-line

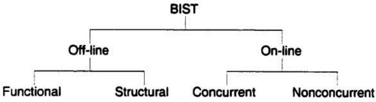


Figure 5.2: BIST methods [8].

BIST, testing occurs during normal functional operating conditions; i.e., the circuit under test (CUT) is not placed in test mode where normal functional operation is locked out. Concurrent on-line BIST is a form of testing that occurs simultaneously with normal functional operation. Concurrent on-line BIST uses redundancy for testing. Redundancy can be achieved by either using coding techniques or duplication and comparison. An example for coding techniques is the parity bit added to check the correctness of the data transmitted over a system bus. In the duplication and comparison approach, the CUT is replicated and a checker circuit is used to take the final decision which is based on the majority decision given by the replicas. In nonconcurrent on-line BIST, testing is carried out while a system is in an idle state. The test process can be interrupted at any time so that normal operation can resume. Testing in nonconcurrent on-line BIST is often accomplished by executing diagnostic software routines or diagnostic firmware routines.

Off-line BIST deals with testing a system when it is not carrying out its normal functions. Functional off-line BIST deals with the execution of a test based on a functional description of the CUT and often employs a functional or high-level fault model. Such a test is implemented as diagnostic software or firmware. Structural off-line BIST deals with the execution of a test based on the structure of the CUT. Fault coverage is based on detecting structural faults.

On-line BIST methods offer two advantages over off-line BIST methods. In on-line testing, the main function of the CUT is not interrupted as is the case in off-line BIST during the test application. Also, the diagnostic information becomes continuously available. However, on-line BIST methods suffer from two main drawbacks [76]. Firstly, the amount of hardware complexity added is, on the average, higher than for off-line BIST methods. In some cases, the added complexity may reach up to 50% of the total area of the IC. Secondly, the design effort is larger than off-line BIST methods because caution has to be taken not to alter the state or function of the CUT. Indeed, a large design effort translates to a longer design phase which is not preferred as the time-to-market is a crucial factor in the VLSI design process. One advantage of using structural off-line BIST method is the flexibility of the design process. The designer's effort is reduced to selecting the approach and the stimulation strategy that fits the CUT and to apply them automatically. In other words, the design process is more standardized than on-line BIST methods, where the design process is significantly influenced by the architecture of the CUT.

To summarize, both on-line and off-line BIST methods have pros and cons, and choosing the proper method is always a trade-off between extra hardware and test application time. However, to achieve a good level of testability a hybrid of both techniques can be used. Accordingly, in this dissertation a testing mechanism, which incorporates both methods, is proposed. We use a combination of a structural off-line BIST method and a nonconcurrent on-line BIST method. Before we present that testing mechanism, we first discuss some issues regarding the structural off-line BIST method.

5.3.2 Structural Off-Line Architectures and Stimulus Structures

The fault model which is assumed by structural testing methods [8] is the well-known single permanent stuck-at fault model. The model assumes that all the components of the CUT are fault free and faults occurs only in the connecting nets. If a fault occurs in one of these nets, then it has a fixed logic value v ($v \in \{0, 1\}$) and denoted as s-a-v. The testing mechanism generates the stimuli to expose these expected faults by assuming that only one fault exists in the CUT. In other words, each stimulus is selected to expose only a single stuck-at fault in the CUT. The single stuck-at fault model ignores other types of faults such as: 1) multiple stuck-at faults, 2) stuck-at open faults 3) bridging faults where two or more nets are shorted, and 4) delay faults, especially if the test is carried out at lower speed than the normal operating speed [76]. That means there is a possibility that a testing mechanism that employs single stuck-at fault model may fail in detecting a fault. However, experience has demonstrated the validity and the efficiency of the single stuck-at fault model due to the following reasons:

- some faults which are not assumed in the fault model may be covered with the used stimuli; and
- detecting delay faults can be achieved to a large extent if BIST mechanism is used, because the testing mechanism can run at the normal operating speed of the CUT.

The architecture of a structural BIST system should contain three main components: 1) a stimulus structure which generates the test stimuli, 2) the non-BIST mode circuitry or CUT, and 3) an output response analyzer (ORA) which produces the final decision as whether the CUT is fault-free or not. These three components

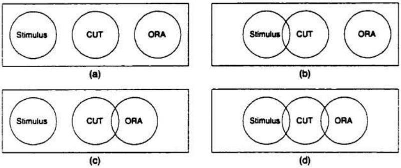


Figure 5.3: Different structural BIST architectures [9].

can be totally separate or merged. Figure 5.3 depicts the different architectures of a structural BIST system. In the merged architectures (parts b, c, and d of Figure 5.3), savings in the hardware complexity of the whole system is realized due to sharing of the memory elements located in the CUT with the stimulus and the ORA circuits. However, in some cases this is not feasible due to some optimization considerations that are imposed by the system requirements.

There are four approaches for designing the stimulus structures [9]:

exhaustive, all possible input stimuli are applied to the CUT. Since the amount of time spent on testing is limited, the number of circuit inputs that can be exhaustively stimulated is usually limited to $\lfloor \log_2(\text{maximum testing cycles}) \rfloor$.

pseudoexhaustive, where the circuit can be partitioned into subcircuits, each dependent on a subset of the inputs, and thus can be stimulated exhaustively. The circuit is tested by testing the partitions.

random, a stimulus source with a fixed sequence is used to excite the CUT. The major advantage of this approach is the low cost of the stimulus structures. Its major disadvantage is the difficulty of designing a stimulus structure that produces all the stimuli required to expose the faults. The stimulus structure can be modeled by a

Approach	Exhaustive	Pseudoexhaustive	Random	Deterministic
CUT Type	n-input CL	CL where the inputs can be grouped	CL, SL	CL, SL
CUT Analysis	none	structural analysis to assign groupings	structural analysis for some methods	structural and ATPG
Stimulus Structure	separate	separate	separate, merged	separate, merged
Test Effectiveness	100% for CL faults	100% for CL faults	requires fault simulation	may require fault simulation
Test Length	2^n	depends on the grouping method	depends on fault simulation results	depends on generation method

Table 5.1: Stimulus design approaches [9]

finite state machine. Thus the designer is usually restricted to choosing a starting state and the number of test cycles. Fault simulation is required to determine which faults are exposed.

deterministic, the stimulus source is required to generate a specific set of stimuli. Analysis of the CUT determines the stimulus set. The structures used for a deterministic approach are usually based on finite state machines.

Table 5.1 depicts the design considerations for the above discussed approaches, where CL stands for combinational logic, SL stands for sequential logic, and ATPG stands for automatic test pattern generation. In the next section we introduce the chip architecture and description of the testing mechanism we are proposing in this dissertation.

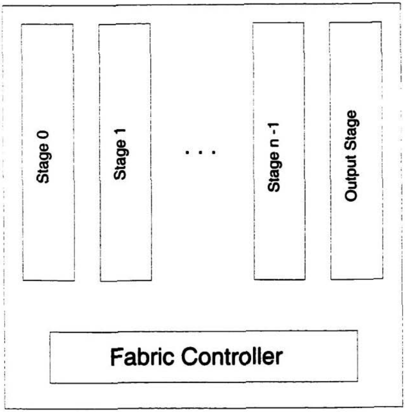


Figure 5.4: Top level description of the BG network.

5.4 Chip Architecture

The BG network is a MIN that enjoys a modular architecture. The modular architecture made the BG network attractive for VLSI implementation. A top level description of the network architecture is depicted in Figure 5.4. There are four main components in the design of the BG network. These components are 1) a 1 × 4 multiplexer used as the building block for the first stage ($Stage_0$), 2) a 4 × 4 crossbar used as the building block for the next $n-1$ stages, 3) a 4 × 1 multiplexer used as the building block for the output stage, and 4) the network main controller (NMC). We denote the 1×4 demultiplexer as 1×4 SE, 4×4 crossbar as 4×4 SE, and the 4×1 multiplexer as OPC to follow the notation we use in this dissertation. We did not carry out the design of the IPC due to time constraints.

We tried to automate the design process by reducing the amount of effort needed when changing the features of network. This is achieved by only changing a few parameters in a header file of the VHDL description. These parameters are the network size N and the output buffer size B_{out}. However, each component in the system still has to be tested and verified separately.

The design of the BG network is restricted to circuits based on synchronous logic rather than circuits based on asynchronous logic. This is to eliminate the problem of logic hazards which makes the function of asynchronous circuits impossible. Logic hazards can be eliminated by using redundant logic but then the circuit becomes untestable. Several other reasons that favor synchronous circuits over asynchronous ones can be found in [8, 77, 78].

The testing mechanism we are proposing, as we mentioned in Section 5.3.2, incorporates both a structural off-line BIST method and a nonconcurrent on-line BIST method. In fact, using only one of these methods to reach a good testability level is either expensive or time consuming. If we use only a structural off-line BIST method, the network operation has to be stalled during the test procedure, causing disruption to the whole system. Also, if we only use the nonconcurrent on-line BIST method, the process of diagnosing and correcting the fault is very expensive in terms of the design effort. Accordingly, we use the nonconcurrent on-line BIST method to regularly check the system for any functional error. If an error is detected, the structural off-line BIST method is activated to check the network for faults. If a fault is detected, the structural off-line BIST method diagnoses it and reconfigures the network to tolerate that fault. The nonconcurrent on-line BIST method is not covered in this dissertation but we will discuss the structural off-line BIST method in more detail.

Before continuing our discussion, we review some issues that are related to the architecture of the network. The switching mechanism of the architecture we are proposing adopts a backpressure strategy. A switching cycle is composed of two periods, a reservation period (sometimes we call it the routing period) and a relaying period. In the reservation period, each IPC sends an internal header representing the cell at the HOL of the buffer that belongs to that IPC. The IPC creates this internal header during the translation of the VPI and VCI in the 5-byte header of the cell. This internal header contains the destination address and priority of the cell to be routed. The structure of this internal header will be explained later. The routing unit in each SE uses the arriving headers to set up a routing pattern for the main cells to be relayed. This process continues until these internal headers reach the OPCs. The output ports acknowledge the arriving headers. A header is positively acknowledged if 1) it successfully reaches its output destination and 2) there is room for the cell represented by that header in the buffer of its destination. The acknowledgments are returned to the IPCs by the switching mechanism. In the relaying period, the IPCs pass the bodies of the cells that were positively acknowledged during the reservation period.

We defined three types of cells, as shown in Table 5.2. Accordingly, the size of the internal header is $n+2$ bits. Recall that n represents the number of the stages in the BG network. Figure 5.5 depicts the structure of the internal header, where D is the address of the destination.

Although a BG network of a single data plane can fulfill up to four broadcast requests as depicted in Figure 5.6, we decided not to include broadcast features in our design because of the added complexity to both the SEs and IPCs. Firstly, the

P_1P_0	Cell type
01	High Priority
10	Low Priority
00	Idle

Table 5.2: Defined cell types.

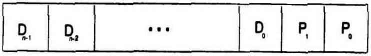


Figure 5.5: Structure of the internal cell header.

complexity of the SEs increases because the routing unit in each SE has to account for the newly added priority. Secondly, the complexity of the IPCs increases because they have to coordinate between each other. This translates to extra communication channels and control units added to the circuitry of each IPC. This coordination is essential when more than four broadcast requests are issued at the main inputs of the network. For example, assume the case when one extra broadcast request is issued at the main input '5' of the network in addition to the four requests shown in Figure 5.6. This request will reach $SE_{4,2}$ on IL_0, $SE_{2,2}$ on IL_2, $SE_{1,2}$ on IL_1, and $SE_{7,2}$ on IL_3 of $Stage_2$. Each SE of that group receives 3 requests. Since the routing decision is the same in all SEs that belong to the same stage and the request at the main input '5' is arriving at different input links of the SEs in $Stage_2$, the request arriving at the main input '5' will be treated differently in these SEs. This translates to the fact that some of these SEs may route and the others may not route the request coming from the main input '5'. In that case, the switching mechanism becomes very complicated in order to account for these diverse actions and to acknowledge back to the IPCs. Additionally, the IPCs have to keep track of the outputs that the broadcasted cells fail to reach and try to reroute these cells again to these outputs in subsequent switching cycles. Indeed, this process demands very complex algorithms which add more hardware to the system and may cause it to run at lower speeds. Accordingly, coordination is needed amongst the IPCs to minimize the complexity.

We believe a design which features broadcasting at an acceptable cost for the BG network is achieved by employing one of the following solutions. The first solution [79] is to precede the BG network with a copy network which generates N copies of each broadcasted cell with each copy destining to one output port. We prefer this solution, because it is suitable for the case of multicasting where a cell is only destining to a subset of the output ports. The second solution is to coordinate amongst the IPCs as we mentioned above.

5.5 System Components

5.5.1 Switching Element

Figure 5.7 depicts the architecture of an 4×4 SE. The architecture of the 1×4 SE is a smaller version of the 4×4 SE. The differences between the two architectures will be highlighted as we proceed in our discussion. The architecture of the SE is divided into the following:

Input Buffer

The input buffer primarily holds the few header bits needed to take the routing decision in the SE. The buffer is composed of one queue in the case of 1×4 SEs and four queues in the case of 4×4 SEs. Each queue is a shift register of size $n+2-i$ bits, where i is the stage order of the current SE. Notice that we do not need to save the whole address of the destination D in the internal header as we advance downstream.

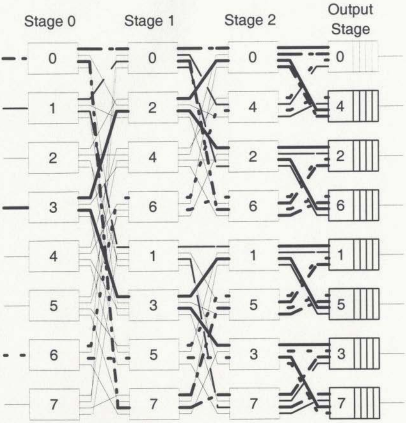


Figure 5.6: Broadcast requests at inputs 0, 1, 3, and 6 are fulfilled.

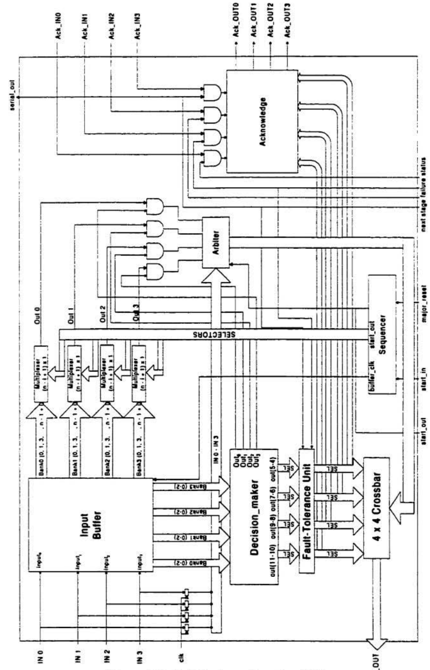


Figure 5.7: Architecture of an 4 × 4 SE.

That is, each SE can skip the bit just after the priority bits (P_1P_0) when passing the headers to the SE of the next stage. If we use the full size of the internal header ($n+2$ bits), we would require a number of cycles ($cycles_{full}$) given by:

$$cycles_{full} = (n+1)(n+2), \tag{5.1}$$

to shift the headers through the network. While, in our approach we only need a number of cycles ($cycle_{used}$) given by:

$$\begin{aligned} cycles_{used} &= \sum_{i=0}^{n}(n+2-i), \\ &= (n+1)(n+2) - \sum_{i=0}^{n} i, \\ &= (n+1)(n+2) - \frac{n(n+1)}{2}. \end{aligned} \tag{5.2}$$

This means we have saved a total of $\frac{n(n+1)}{2}$ cycles in the reservation period. A block diagram of the input buffer of an SE at $Stage_i$ is shown in Figure 5.8.

Pushout Multiplexers

After the headers are received from the previous stage and the routing pattern is established within each SE, the pushout multiplexers are used to push the header bits to the SEs of the next stage. Each queue in the input buffer is served by an $n-1+i \rightarrow 1$ pushout multiplexer. The select lines for these multiplexers are fed from the sequencer, which we will describe later.

Decision Maker

The decision maker is entirely a combinational logic circuit. It uses the first three bits (0-2) from each queue in the input buffer to take the routing decision. That is, the decision maker has 12 inputs in a 4 × 4 SE. While, the number of the outputs in an 4 × 4 SE is 12 bits and they are divided as follows:

- Four pairs (S_0, ..., S_3) are used as selectors to decide which input link will be

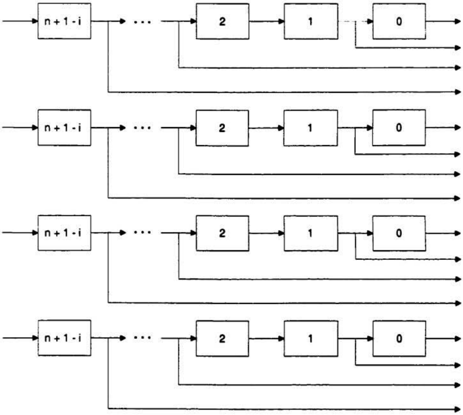


Figure 5.8: Block diagram of the input buffer bank of an SE at $Stage_i$.

connected to which output link. The S_i pair is used to decide which input link should be connected to OL_i, for $0 \leq i \leq 3$.

- The other four outputs are used to identify which inputs have received active cells and which ones have received idle cells. This is needed when shifting the headers to the next stage.

We have discovered that the Synopsys synthesis tool does not generate good minimization levels for large combinational circuits such as the one we are dealing with here. Accordingly, we decided to use the well-known combinational logic minimization tool (Espresso) developed at University of California, Berkeley [80, 81]. The following algorithm was followed because we also discovered better minimization results are obtained if the problem is divided into smaller ones:

1. Find Espresso minimization for the truth table of the 12-inputs and a subset of i outputs from the total 12 outputs.

2. Repeat the above process until all combinations C_i^{12} are exhausted.

3. Repeat both steps 1 and 2 for all values of i, where $1 \leq i \leq 12$.

We then arrange the results obtained in an ascending order for each value of i. Afterwards, we identify the best set of combinations that cover all outputs and results in the minimum hardware complexity. By trial and error, the set is found to contain two groups; $\{11, 10, 9, 8\}$ and $\{7, 6, 5, 4, 3, 2, 1, 0\}$. The latter group represents the $S_0, \ldots, S_3$ pairs and the former represents the outputs used to identify the active and idle arriving cells. That implies that there is certain amount of correlation among the selector bits $\{7, 6, 5, 4, 3, 2, 1, 0\}$. Similarly, the acknowledgment bits $\{11, 10, 9, 8\}$ have a certain amount of correlation.

Arbiter

The arbiter selects between the main inputs of the SE and the outputs of the buffer bank. In the reservation period, the arbiter selects the outputs of the pushout multiplexers to route the headers to the next stage. In the relaying period, the arbiter selects the main inputs of the SE.

Fault Tolerating Unit

Notice that in Figure 5.7 the selection pairs S_0 to S_3 from the decision maker are feeding the fault tolerating unit. In case of faults in the SEs of the next stage, the fault tolerating unit modifies the select lines by giving priority to the cells routed to OL_0 and OL_2 because high priority cells are usually routed through these links. For example, assume the SE in the next stage that is connected to OL_0 is faulty. The fault tolerating unit will always redirect the cells targeting OL_0 to OL_1. The cells that are destining to OL_1 will be negatively acknowledged, by the acknowledgment unit we will describe later, and will be tried in the next switching cycle. In case of no faults in the next stage SEs, the fault tolerating unit acts transparently so that the input select lines are transmitted to the outputs of the fault tolerating unit without any modification. The faulty SEs of the next stage are reported to the current SE through the "*next stage failure status*" word fed from the BIST circuitry of the SE.

Selection Unit

The selection unit is the true crossbar part in the SE. Its function is to switch the arriving cells to their requested output links. The switching decision is passed to the selection unit through the modified select lines originating from the fault tolerating unit.

Acknowledgment Unit

The acknowledgment unit performs the reverse function of the selection unit. It receives the acknowledgments from the next stage and routes them back to the previous stage. Notice that the select lines that are used to feed the selection unit are the same ones that are feeding the acknowledgment unit. The reason is that the acknowledgment unit has to be offered the final routing decision to give the proper acknowledgment. Notice also that the select lines are ANDed with the next stage failure status word to block a positive acknowledgment that might arrive from the faulty SEs in the next stage.

Sequencer

The sequencer plays a very important role in the operation of the SE, as well as in the operation of the whole network. The sequencer synchronizes the process of moving the header from one SE to another. Once it receives a negative edge on the triggering signal (START_IN) from an SE in the previous stage, denoting the start of sending the header bits, the sequencer activates the clock of the input buffer to receive the header bits from the previous stage. It also inserts one wait clock cycle until the routing decision settles within the decision maker. Then, it alarms the SEs of the next stage by a negative edge on a triggering signal (START_OUT) similar to the one it previously received from the previous stage. Then it iterates $n - i + 1$ cycles through the select lines fed to the pushout multiplexers discussed previously. Figure 5.9 depicts the ASM chart describing the sequencer function.

Testing Unit

The testing unit is not depicted in Figure 5.7 for simplicity. The testing unit represents the off-line structural BIST part of the testing mechanism we are proposing. It is obvious that the off-line structural BIST method is distributed throughout the network, since every SE has its own testing unit. We decided to follow the strategy depicted in Figure 5.3.a so that we do not complicate the structure of the internal core of the SE. A complex core translates to a slower system speed, and speed is critical since we are targeting a system speed of 155.52 Mbps (OC-3c). We also followed the random approach for test stimuli design discussed in Section 5.3.2 because of its

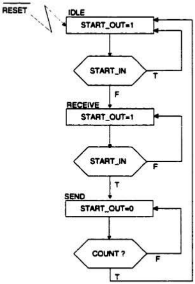


Figure 5.9: ASM chart of the SE sequencer.

simplicity and cost effectiveness in the case of our design. The fault coverage ranged from 85% for 4 × 4 SE to 92% for 1 × 4 SEs.

During the normal mode of operation, the testing unit acts transparently such that the core of the SE is connected to all other SEs in the network. During the testing mode, the testing unit isolates the core from the other parts of the network. Figure 5.10 depicts the block diagram of the testing unit during the test period. The linear feedback shift register (LFSR) represents the stimulus circuit and both the multiple in shift register (MISR) and signature analyzer represent the ORA. The other blocks are used to test the links between the current SE and the other SEs in both the previous and the next stages. The downstream failure status buffer holds the status of the SEs in the next stage. The status word out of this buffer is used to feed the acknowledgment and fault tolerance units of the current SE. Also, this buffer is serially connected to the SEs surrounding the current SE and exist in the same stage. This serial connection is used during the test period when checking the testing mechanism itself. The testing unit steps through 10 states. The first state is when the network is functioning in its normal mode of operation. Stepping through these states is achieved by the network main controller (NMC). We will list these states in more detail when we discuss the NMC.

Table 5.3 depicts the estimated distribution of the hardware complexity *in gates* for the different SEs in an 16 × 16 network estimated by Synopsys 1998.02-2 *design_analyzer* tool. A gate is a 2-input NAND gate. The results are based on 0.35 μm CMOS technology. The last row in the table represents the hardware complexity of an SE belonging to the first stage. The results in Table 5.3 suggests that the decision maker constitutes about 60% of the total complexity of an 4 × 4 SE in the second stages and above. In fact the maximum speed of the SE is determined by the speed

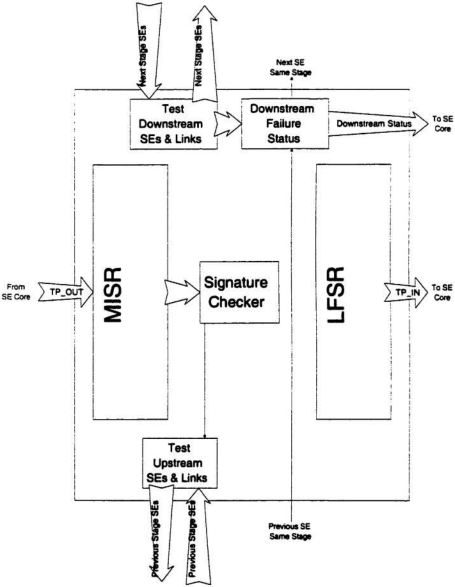


Figure 5.10: Block diagram of the testing unit during test period.

of the decision maker through which the critical path of the SE passes. Table 5.4 shows the delays for the critical paths of the SEs that belong to the 16 × 16 network. The estimated delay through the decision maker is 2.53 ns which roughly represents 44.5% of the total delay of the critical paths in 4 × 4 SEs. Clearly, this is a very high percentage. Luckily, the effect of the critical path of the decision maker is felt only during the routing period of the network. To reduce the effect of the critical path in a 4 × 4 SE the sequencer adds an extra wait clock cycle before routing the bits of the internal header to the next stage. That way we add extra latency to the system. However, the ratio of the number of the added clock cycles (n) to the total number of clock cycles in one complete switching cycle is insignificant. We will present the total number of clock cycles in one switching cycle after discussing the other components.

5.5.2 Output Port Controller

The design of the OPC we propose here is composed of one single buffer queue. That is, we do not assume a scheduling mechanism of multiple service queues. As depicted in Figure 5.11, the OPC has the following blocks:

Input Buffer

The input buffer has the same function as the input buffer bank of the SEs. However, the input buffer in the OPC holds only the priority bits (P_1P_0), which are two bits/input. The input buffer has the same architecture as the input buffers used in the SEs, except it does not contain any pushout multiplexers. This is because we do not need to push the header bits to any subsequent stage.

Sequencer

The sequencer of the OPC has nearly the same architecture as the sequencer used in the SEs. However, it adds extra delay cycles between the routing period and the relaying period. These delay cycles are needed until the acknowledgment decision is

	Buffer Bank		Sequencer		Decision Maker		Arbiter + fault Tolerance + Acknowledgment + Crossbar + Others		Testing		Total	
	Comb.	Seq.	Comb.	Seq.	Comb.	Seq.	Comb.	Seq.	Comb.	Seq.	Comb.	Seq.
Stage$_1$	16.68	107.80	16.22	33.53	1007.86	0	177.32	21.08	190.92	195.39	1409.00	357.80
Stage$_2$	20.67	86.95	16.98	33.53	1007.86	0	184.88	21.08	190.92	195.39	1421.31	336.95
Stage$_3$	13.04	65.63	13.80	27.78	1007.86	0	161.55	21.08	190.92	195.39	1387.17	309.88
					Comb.		Seq.					
Stage$_0$	0	33.05	15.66	38.32	45.27		5.51		139.68	152.14	200.61	229.02

Table 5.3: Distribution of the hardware complexity of the SEs in a 16 × 16 (in gates).

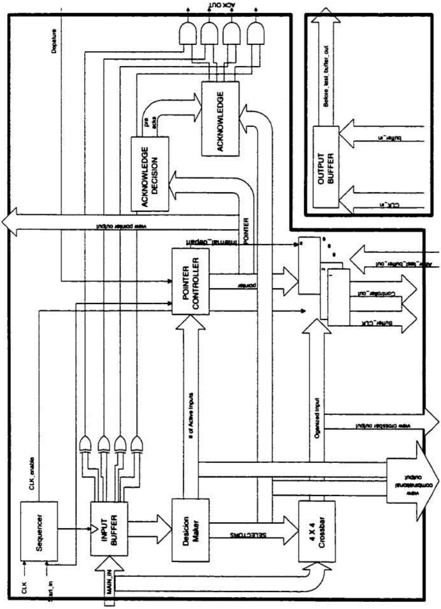


Figure 5.11: Architecture of the OPC.

	Delay (ns)
$Stage_0$	3.98
$Stage_1$	5.57
$Stage_2$	5.72
$Stage_3$	5.58

Table 5.4: Delays of the SEs critical paths.

transferred to the IPCs and the first bit of the cell is relayed across the network until it reaches the OP. The number of these delay cycles ($cycles_{delay}$) is given by:

$$cycles_{delay} = 2 + 2 \times n. \tag{5.3}$$

The first two cycles are required until the logic settles in the OPC, the next n cycles are required until the acknowledgment reaches the IPCs, and the last n cycles are required until the first bit of the relayed cells reaches the OPCs. Figure 5.12 shows the ASM chart of the sequencer.

Sorter

During the routing period, the sorter is responsible for sorting the arriving cells in a descending order based on their priorities. This is essential for two reasons: 1) to ease the process of placing the cells into the buffer, and 2) to ease the process of identifying idle cells from active ones. The sorter also provides the pointer controller with the number of active arriving cells. The sorter is composed of two combinational circuits. The first circuit is an 8-input 11-output circuit called the decision maker. Eight of the 11 outputs are used to feed a 4×4 crossbar network, which represents the second circuit. The other three output bits are used to hold the count of the arriving cells ($0 \rightarrow 4$). These three bits are feeding the pointer controller which will be described in the next section. The crossbar passes the arriving cells in a descending order to the buffer controllers.

Pointer Controller

The pointer controller provides a pointer that points at the next empty location in the output buffer. This value is used by the buffer controllers and the acknowledgment decision unit. The value of the pointer for the next switching cycle ($pointer(t+1)$) is given by:

$$pointer(t+1) = pointer(t) + arrival(t) - departure(t+1); \qquad 0 \leq pointer(t) \leq B_{out}+1, \tag{5.4}$$

where $pointer(t)$ and $arrival(t)$ are the value of the pointer and the number of arriving cells at the end of the previous cycle, respectively, $departure(t+1)$ is the indication of whether a cell will depart from the buffer or not in the current switching cycle, and B_{out} is the size of the buffer in the OPC. Notice that at most one cell is allowed to depart from the OPC in any switching cycle, i.e. $departure(t) = \{0, 1\}$. When the value of $pointer(t)$ is 0 it means the buffer is totally empty, and when it is $B_{out} + 1$ it means the buffer is totally full and no cells could be placed in the buffer.

Acknowledgment

The acknowledgment process goes through three steps in the OPC. The first step is carried out by the block "Acknowledge Decision" in Figure 5.11. In that step an initial estimate of the acknowledgment word is formed based on the buffer occupancy and the pointer value. This initial acknowledgment is meant for the sorted cells. Accordingly, in the second step of the acknowledgment process the "Acknowledge" block shown in Figure 5.11 redirects this initial acknowledgment to its proper order of the arriving cells. The Acknowledge block has the same structure as the acknowledgment unit of the SE. In the third step, a correction has to be made to the output from the acknowledge block because the initial acknowledgment is not based on the status of the cells, whether idle or active. This correction is made by ANDing the status of each cell with its corresponding output from the acknowledge block.

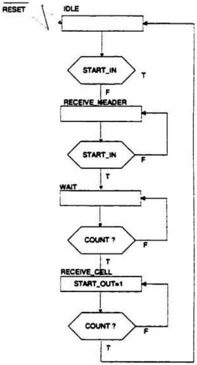


Figure 5.12: ASM chart of the OPC sequencer.

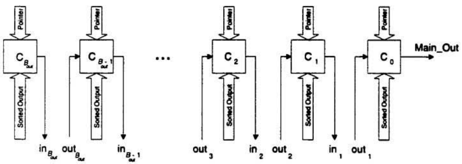


Figure 5.13: Block diagram of the Buffer controllers.

Buffer Controllers

The number of the buffer controllers is $B_{out} + 1$. Each buffer controller connects two successive locations in the output buffer. Figure 5.13 depicts a block diagram of the buffer controllers. All controllers are fed from the pointer controller and the four sorted outputs from the 4 × 4 crossbar. If the value $(i - pointer)$ is negative — where i represents the location of controller C_i —, then location i in the buffer is occupied. While, if that value is positive and less than 4, then the controller places cell $(i - pointer)$ from the crossbar at location i of the buffer. That way we can simultaneously write to 4 sequential locations of the buffer in the same switching cycle. If the output buffer is empty and *departure* = 1, the current design allows a fresh arriving cell to depart from the OPC without placing it in the first location of the buffer. This reduces the latency in the buffering mechanism of the OPC and thus the whole network.

Notice that all the buffer controllers are identical, except controllers C_0 and $C_{B_{out}}$. C_0 connects location 1 of the buffer to the outside world. $C_{B_{out}}$ is not fed from any buffers because it is located at the end of the queue. All other controllers are fed from one location and feed to the next location in the buffer queue. That is controller C_i

is fed from the output of location $i + 1$ and feeds to the input of location i.

Buffer

The buffer is composed of B_{out} banks. Each bank is a basic shift register equal to the size of the cell. It is connected to two controllers as described above in Figure 5.13. Both the sizes of the cell and the output buffer B_{out} are tuned in the parameter file we mentioned earlier in Section 5.4.

Testing Unit

The testing unit of the OPC is similar to the testing unit of the SE. Accordingly, here too the earlier discussion for the testing unit of the SE is applicable. However, the testing unit of the OPC has the extra function of testing the buffer. Testing the buffer is achieved by pushing a sequence of 0s followed by a sequence of 1s and checking whether the output of the buffer is following these sequences. Notice that in Figure 5.11, the buffer is shown separated from the buffer controllers as the testing unit is inserted between both. The testing unit of the OPC also acts transparently in the normal mode of operation of the network.

Table 5.5 depicts the distribution of the hardware complexity for the OPC. The complexity of the OPC is marginally dependent on the network size N, where N only affects the number of delay cycles needed between the arrival of the header and the arrival of the first bit of the cell. The results in Table 5.5 are based on a buffer size of 6 cells and a cell size of 2 bytes.

5.5.3 Network Main Controller

The NMC plays a vital role in the operation of the BG network. The ASM chart describing the functioning of the NMC is depicted in Figure 5.14. The network can operate either in a normal mode or a testing mode. In the normal mode, the NMC

	Seq.	Comb.
Decision Maker	–	231.41
Sequencer	40.98	38.59
Pointer Controller	58.7	77.71
Input Buffer	44.07	–
Acknowledge	–	42.95
Crossbar	–	23.63
Buffer Controllers	–	79.48
Buffer	392.14	–
Testing Unit	373.39	228.76
Others	–	90.52
Total	909.27	813.05

Table 5.5: Distribution of hardware complexity for OPC (in gates).

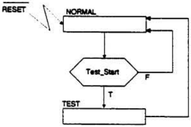


Figure 5.14: The ASM chart of the network main controller.

initiates the routing period of the SEs that belongs to $Stage_0$ by bringing all the start_in signals of these SEs to logic 0. The NMC keeps these signals low for $n+2$ cycles, which is the number of cycles needed to push the internal headers to the SEs of $Stage_0$. Then, the NMC brings all the start_in signals again to logic high and waits a number of cycles that are needed to continue the routing period. The total number of clock cycles in each switching cycle ($cycles_{switching}$) is given by:

$$cycles_{switching} = (n+1)(n+2) - \frac{n(n+1)}{2} + 2 + 2 \times n + 8 \times cell\_size. \qquad (5.5)$$

where *cell_size* is in bytes. The network operates in the normal mode indefinitely as long as the input signal test_start to the NMC is not activated. When it is activated the NMC changes to the testing mode. In the testing mode, the NMC activates the testing units in the SEs and OPCs. The NMC also enables these testing units to step through 10 states. During each state part of the testing protocol is executed. The states of the testing unit in the SE are:

Isolate: the testing units isolate the cores of the SEs and the OPCs from all other components.

Check_Mechanism/Check_Buffer: for the SEs, this state is called check_mechanism and for the OPCs it is called check_buffer. During this period a sequence of 1s, followed by a sequence 0s, is shifted through the testing mechanism to check the serial links between the testing units for any faults. The output sequence is collected by the NMC and checked for faults. For an OPC, the testing unit shifts sequences of 1s and 0s through the memory elements of the buffer and checks the output sequence for faults.

Check_links_1: the first state to check the links for s-a-1 faults. Each testing unit sends 0s across the all output links and receives 0s over all inputs links.

Check_links_2: used to latch the results of the previous test state.

Check_links_3: similar to state check_links_1 but for s-a-0 faults.

Check_links_4: similar to check_links_2. By the end of this test state, each SE holds the status of the links to the next stage.

Check_structure_1: the testing units prepare for structural testing of the SEs and the OPCs by hooking up the LFSRs and the MISRs to the internal cores.

Check_structure_2: carrying out the structural testing.

Check_structure_3: latching the results of the previous test state in the next stage failure status word of each SE.

Shift_test_results: a copy of the test results are serially shifted to the outside world through the NMC. After this test state, the NMC (sets) resets an output signal "test_result" if (no) faults exist.

The total complexity of the NMC is 531.81 gates; 391.44 combinational and 140.37 sequential. The critical path is found to be 4.35 ns. Notice that:

- the network can tolerate a fault in an SE by using the fault tolerance units of the SEs located in the previous stage and connected to the faulty SE. Recall that the fault tolerance unit of an SE redirects the traffic through the output links depending on the next stage fault status word set up in the check_structure_3 test state above.

- the network can not tolerate faults in the OPCs and the 1 × 4 SEs, but it can detect and locate them.

In the next chapter we will discuss the fault tolerance properties of the BG network.

	P_0	P_1	D_2	D_3
IL_0	0	1	1	1
IL_1	0	1	1	1
IL_2	1	0	1	0
IL_3	0	0	0	0
in Hex	2	C	E	C

Table 5.6: Illustration of cell arrival in Figure 5.15.

5.6 Simulation and Test Results

Verification of the network has been carried out on different levels of the design hierarchy. Testbenches were used to check the functionality of the 1 × 4 SEs, 4 × 4 SEs, OPCs, NMC, and the whole design. We wrote C++ routines to generate the test vectors needed to test each component. Figure 5.15 shows the simulation results for a synthesized 4 × 4 SE in $Stage_2$ using 0.35 μm CMOS technology. The shown snapshot represents the routing period of the SE, which is the critical period due to the involvement of the decision maker. The negative edge on the start_in signal from the previous stage at time stamp 44935 ns denotes the start of the header arrival from the previous stage. There are three cells arriving in that routing period at IL_0, IL_1, and IL_2. As depicted in Table 5.6, the headers arriving at IL_0 and IL_1 have low priority and the cell arriving at IL_2 has high priority. All the arriving headers have a routing bit equal to 1. Accordingly, the cell arriving at IL_3 will be directed to OL_2 and one of the other headers will be routed to OL_3. Also, idle cells are pushed through OL_0 and OL_1. The resulting sequence on the output links MAIN_OUT of the SE, as shown in Figure 5.15, is 2, 1, and 1. The results of the timing diagram suggests that the SE can run comfortably at a speed of 200 MHz, which is sufficient for our target speed 155.52 MHz.

Figure 5.16 describes the operation of the OPC. As shown in Figure 5.16 and

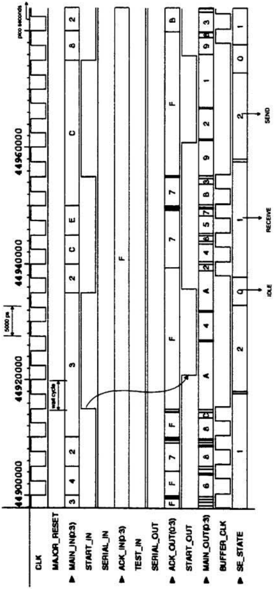


Figure 5.15: Simulation results for an 4 × 4 SE.

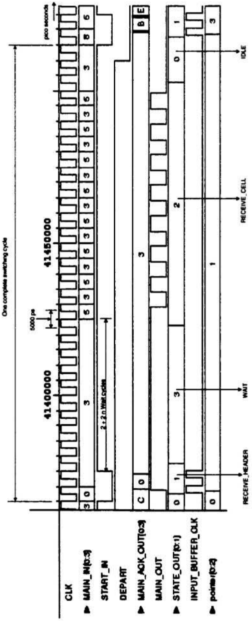


Figure 5.16: Simulation results for an OPC.

	P_0	P_1
IL_0	0	0
IL_1	0	0
IL_2	0	1
IL_3	0	1
in Hex	0	3

Table 5.7: Illustration of cell arrival in Figure 5.16.

illustrated in Table 5.7, there are two low priority cells arriving at IL_2 and IL_3 in the current switching cycle. The pointer value is equal to 1 in the current switching cycle because the previous value of the pointer is 0, two cells were arriving in the previous switching cycle (not shown in Figure 5.16) and the current departure is 1 (refer to Equation 5.4). The clock speed in the timing diagram is 200 MHz.

The timing diagram describing the operation of the NMC as well as that for the whole network are not provided because they require huge space for presentation, especially during the the testing mode of operation. However, the obtained simulation results have demonstrated the correct functionality of the network.

5.7 Summary

We discussed the design of the BG network. The design is mainly built from three main modules; the switching element (SE), the output port controller (OPC), and the network main controller (NMC). The architecture and the function of each module were described in detail. Complexity and timing requirements for each module were provided. The design of the BG network features a BIST mechanism which consists of two methods. The first is an off-line structural method which is distributed all over the network because each module, except the NMC, has its own internal structural testing unit. The second is an on-line nonconcurrent method, which it is used at the system level. We discussed the test protocol used for the proposed testing mechanism. We also provided the simulation results for the SE and the OPC. The results suggested that the network can operate comfortably at a speed of 200 MHz. This means that the network can comfortably support the OC-3 bit rate which is 155.52 Mbps. In the next chapter, we discuss the fault tolerance properties of the BG network.

Chapter 6

Fault Tolerance and Reliability Properties

6.1 Introduction

In this chapter, we discuss the fault tolerance and the reliability performance of the BG network. The fault tolerance properties of the BG network are shown to be exceptionally better compared to other MINs. The network is a single fault-tolerant and robust in presence of multiple faults of the switching elements. We use three models to evaluate the terminal reliability, the broadcast reliability and the network reliability of the BG network. We also use realistic failure rates for the network building blocks. These failure rates are based on a 0.8 μm BiCMOS implementation. Throughout this chapter we assume that the interstage links connecting the SEs are highly reliable.

6.2 Background

In [82], a discussion of eleven fault-tolerant MINs has been presented. A general criterion to compare the fault-tolerant characteristics of different MINs was described. In addition, a concluding discussion on the different techniques used to obtain a fault-tolerant structure was presented. These techniques were divided into two groups:

techniques that alter the topology and techniques that do not alter the topology. The former group of techniques are used in multipath MINs, where other paths are used if a fault is detected in one of the paths. Replication and adding extra stages are examples for the latter group, i.e., trying to increase the number of paths between input-output pairs.

The reliability is another important aspect used to differentiate MINs. Several studies on the subject of MINs' reliabilities have been reported [83]-[88], [16]. The metrics used to measure the reliability of MINs are terminal reliability (TR), broadcast reliability (BR), and network reliability (NR). The BG network exhibits superior reliability performance over other MINs that have a similar hardware complexity. For example, a new MIN that has a comparable hardware complexity has been recently reported [89, 90]. However, when compared with the BG network, the performance of the latter network is poorer because it employs almost the same routing technique used in a parallel banyan network. It has been shown that the performance of the BG network is superior to that of parallel banyan network [65].

6.3 Fault Tolerance Properties of the BG Network

A fault-tolerant MIN is one that is able to route cells from input ports to the requested output ports, even when some of its network components are faulty. A fault can be transient or permanent. Here we assume all faults are permanent. To study the fault tolerance properties of a MIN, we have to define a fault tolerance model, fault tolerance criterion and fault tolerance method. The fault tolerance model characterizes all faults assumed to occur, stating the failure modes (if any) for each network component. The fault tolerance criterion is the condition that must be met for the network to be said to have tolerated a given fault or faults. Most studies assume a fault tolerance criterion that satisfies the full access property in MINs. A network is said to have full access property if a cell at any of the input ports can be routed to any of the output ports. The fault tolerance method is the way to overcome the faults described in the fault tolerance model in order to fulfill the fault tolerance criterion. The fault tolerance model we use is as follows:

1. SE faults are random and independent.
2. SE faults are permanent.
3. Each faulty SE is totally unusable.
4. All faults are detected and located with full success.
5. Any link failure can be subsumed by the failure of the SE that immediately precedes the link.

The final fault tolerance model of the BG network is achieved by adding the following modifications:

1. Faults can occur in any SE except in the first stage.
2. The output buffers are highly reliable.

Figure 6.1 depicts all the SEs in the first few stages of the BG network that can be visited by a cell originating at input port i. If the cell is routed upwards, i.e. the routing bit is 0, then it may visit either $SE_{i,1}$ or $SE_{i+2,1}$ in $Stage_1$. These two SEs are encapsulated by an ellipse and marked as i and $i+2$, respectively. However, if the cell is routed down, then it may either visit $SE_{i-1,1}$ or $SE_{i+1,1}$ in $Stage_1$. These two SEs are encapsulated by an ellipse and marked as $i-1$ and $i+1$, respectively. Here is an example. $SE_{i+6,2}$ may direct the arriving cell to $SE_{i+6,3}$ or $SE_{i+14,3}$ if the

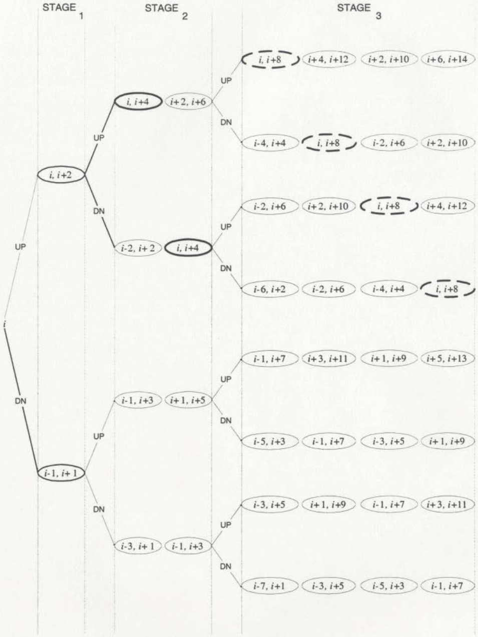


Figure 6.1: All possible SEs that can be visited by a cell arriving at input port i.

routing bit is 0 and may direct the arriving cell to $SE_{i+2,3}$ or $SE_{i+10,3}$ if the routing bit is 1. To summarize, each SE can send the arriving cell at its input to one of 4 SEs in the next stage. Additionally, each encapsulated SE pair in Figure 2 implies the fact that these two SEs are connected to the same SE in the previous stage and a cell existing in that SE may reach one of them. These SEs are called critical pairs. In [91], we have proven that the BG network will lose full access property *iff* any of the critical pairs fails. By inspecting Figure 6.1 we notice that $SE_{i,j}$ forms two critical pairs with $SE_{i+2^j,j}$ and with $SE_{i-2^j,j}$. We denote the critical pair $SE_{i,j}$ and $SE_{i+2^j,j}$ together as $(i, i+2^j, j)$.

More unreachable output ports exist as the failure of critical pairs takes place in the stages close to the input ports. For example, in a 16 × 16 BG network if the pair $(i, i+2, 1)$ fails then all output ports $i-6$, $i-4$, $i-2$, i, $i+2$, $i+4$, $i+6$, $i+8$, $i+10$, $i+12$ and $i+14$ are completely unreachable for a cell originating at input port i. However, if the critical pair $(i, i+4, 2)$ fails then only output ports $i-4$, i, $i+4$, $i+8$ and $i+12$ are unreachable by a cell originating at input port i.

To summarize, we can conclude that the BG network has N critical pairs in each $Stage_m$, where $0 < m < n-1$ and only $N/2$ critical pairs in $Stage_{n-1}$ because in this stage the critical pairs $(i, i-2^{n-1}, n-1)$ and $(i, i+2^{n-1}, n-1)$ are basically the same (we know that $2^{n-1} = N/2$). Additionally, it is clear that the BG network is a single fault-tolerant network because any single fault in the network can be tolerated. Although not all double or multiple SEs faults could be tolerated because certain combinations of SEs form critical pairs. Consequently, the BG network can be considered as a robust network in the case of multiple faults. We also notice that the closer the faulty critical pairs are to the input ports the more the number of output ports that are likely to be unreachable.

	8×8	16×16	32×32	64×64	128×128
$Stage_0$	139200.52	140496.34	141792.70	143088.12	144384.19
$Stage_1$	565982.64	572443.21	580711.44	586881.44	599731.11
$Stage_2$	549153.84	566676.24	572431.76	597945.10	589817.81
$Stage_3$	-	549844.20	567491.35	572375.19	581027.74
$Stage_4$	-	-	550914.6	567984.60	573942.08
$Stage_5$	-	-	-	550914.63	568291.34
$Stage_6$	-	-	-	-	551318.76
OPC	545396.33	558031.68	570245.21	584101.70	598015.23
NMC	151927.71	148280.19	155959.48	153146.04	159938.72

Table 6.1: SEs' complexities (in μm^2) for different sizes of the BG network.

6.4 Reliability Analysis

In [92], a two-fold reliability model for 0.8 μm BiCMOS technology is presented to estimate reliability of VHSIC/VHSIC-like and VLSI integrated circuits. We use this model to estimate the different reliability measures for the BG network. Firstly, the failure rates are evaluated based on the area of each element in the system. These failure rates, represented by the failure rate λ_I, are used to evaluate the reliabilities of the individual SEs. Secondly, contributions to the system reliability from other factors, such as environmental operating conditions, expected number of pins, fabrication quality and packaging, are taken into account. These factors are represented by another failure rate λ_{II}. In our analysis we assume that the interconnections between stages are highly reliable. Table 6.1 emphasizes the design complexity for different network sizes ranging from 8×8 up to 128×128. The reason we use 0.8 μm BiCMOS technology is that the above mentioned model is only applicable to that particular technology. Table 6.2 provides the failure rates of the SEs using the estimated areas in Table 6.1 according to the first part of the model. We notice the ratio between the area of a SE in the first stage to the area of another SE in the other stages is not the same as the ratio between their failure rates. This is because, in the model, the

	8 × 8	16 × 16	32 × 32	64 × 64	128 × 128
$SE_{i,0}$	0.034013266	0.034034986	0.034056716	0.034078429	0.034100154
$SE_{i,1}$	0.041166947	0.041275239	0.04141383	0.041517251	0.041732636
$SE_{i,2}$	0.040884864	0.041178573	0.041275047	0.041702699	0.04156647
$SE_{i,3}$	-	0.040896436	0.041192236	0.041274098	0.041419132
$SE_{i,4}$	-	-	0.040914378	0.041200504	0.041300362
$SE_{i,5}$	-	-	-	0.040914379	0.041205645
$SE_{i,6}$	-	-	-	-	0.040921153
OPC	0.040821881	0.041033674	0.041238396	0.041470657	0.041703874
NMC	0.034226598	0.034165458	0.034294178	0.034247019	0.034360878

Table 6.2: Failures/10^6 hours (λ_I) for the components of 128 × 128 BG network due to the first part of the model.

8 × 8	0.222
16 × 16	0.233
32 × 32	0.268
64 × 64	0.326
128 × 128	0.471

Table 6.3: Estimated failures/10^6 hours (λ_{II}) due to the second part of the model.

significance of the term that contains the area is minor when compared to the other factors that are technology dependent. Table 6.3 provides the failure rates due to the second part for different network sizes of the BG network.

Three measures are normally used to assess the reliability performance of MINs [87]. These are terminal reliability, broadcast reliability, and network reliability.

6.4.1 Terminal Reliability

TR is the probability that there exists at least one fault-free path from a particular input port to a particular output port. TR is always associated with a terminal path (TP) which is one-to-one connection between an input port (the source) and an out port (the destination). A network is considered failed if it is not able to establish a connection from a given source to a given destination. The set of paths in a network between a given input-output pair is represented as a directed graph, sometimes referred to as the redundancy graph (*R-graph*) [93], with its vertices representing the SEs and the edges representing the connecting links. This *R-graph* is used to determine the TR of the network. In our analyses, we assume constant failure rate λ of all the components, hence reliability of the SE is given by:

$$p = e^{-\lambda t}. \tag{6.1}$$

We know that a cell has an option of being routed through either one of two links in each stage. The *R-graph* of the BG network is not the same for each input-output pair and is dependent on the destination. Certain input-output pairs use only a pair of SEs in each stage of their TPs and hence have a lower TR than the other pairs which use more than two SEs at certain stages. The TR of an input-output pair is the lowest when the least significant $\lceil \frac{n}{2} \rceil$ tags bits are identical. Figure 6.2 shows an example of a best case *R-graph* for a 16 × 16 BG network and Figure 6.3 shows another example for a worst case *R-graph*. Figure 6 depicts the worst case *R-graph* for a general $N \times N$ BG network.

We take the worst case to evaluate the TR of the BG network. At each stage one of the critical pair SEs has to be fault free. Therefore, the worst case TR of the BG network is given by:

$$TR = p_0 \times \left[\prod_{i=1}^{n-1} (1 - (1 - p_i)^2) \right] \times p_{OPC} \times p_{NMC}, \tag{6.2}$$

where p_i is the reliability of $SE_{k,i}$ $(0 \leq k \leq N-1)$, p_{OPC} is the reliability of the OPC, and p_{NMC} is the reliability of the NMC. All the p_i, P_{OPC}, and p_{NMC} are evaluated using the values previously presented in Table 4. p_{NMC} is included in Equation 6.2 because if the NMC fails the whole system fails. Table 6.4 depicts the behavior of the TR for various sizes of the BG network. The figures obtained in Table 6.4 signify

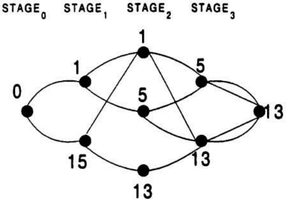


Figure 6.2: *TR* best case *R-graph* for 16 × 16 BG network.

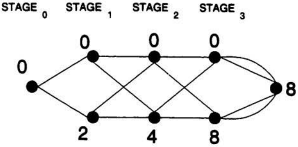


Figure 6.3: *TR* worst case *R-graph* for 16 × 16 BG network.

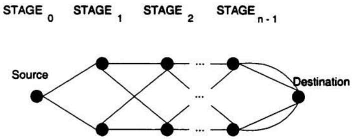


Figure 6.4: *TR* worst case *R-graph* for $N \times N$ BG network.

Mission Time (hours ×1000)	8 × 8	16 × 16	32 × 32	64 × 64	128 × 128
20	0.9978198	0.9978157	0.9978079	0.9978031	0.9977950
40	0.9956417	0.9956321	0.9956152	0.9956042	0.9955868
60	0.9934657	0.9934493	0.9934220	0.9934034	0.9933753
80	0.9912917	0.9912673	0.9912283	0.9912008	0.9911606
100	0.9891200	0.9890861	0.9890340	0.9889964	0.9889429

Table 6.4: Behavior of the *TR* for various sizes of the BG network.

the fact that the TR is not affected by the network size for a mission period of 11 years. This is because the SEs' failure rates are small and close to each other. In fact the estimated failure rates in Table 6.4 does not take into account the other factors mentioned above, which are packaging, number of pins, etc., and represented by the failure rates provided in Table 6.3. We could not incorporate these factors when evaluating the TR because they describe the whole system, not a specific input port or a specific output port. Obviously the results in Table 6.4 demonstrate that the TR of the BG network is very high even for large networks. That is, any particular input-output connection can be established with a high reliability.

6.4.2 Broadcast Reliability

Although we did not incorporate the broadcast feature in the design of the BG network due to the reasons mentioned in Section 5.4, we briefly discuss the BR of the BG network in this section. BR is the probability that there exists at least one fault free path from a particular input port to all output ports. BR is always associated with a broadcast path (BP) which is a connection from one source to all destinations in the network. Under this criterion, the network is considered failed when the connection cannot be made from a given input port to at least one of the output ports. Broadcasting is achieved by routing the broadcast cell to two output links, one is up and the other is down, as it reaches any SE. Another method to implement broadcasting is to generate N copies of the broadcast cell at the input port, each one of these copies destined to a different destination.

Contrary to the TR , the BR is identical for all input ports and therefore the BG network has a uniform BR. The broadcast *R-graph* for an 8×8 network is shown in

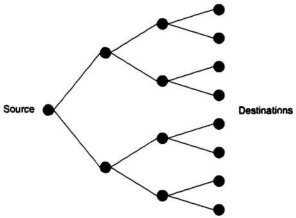


Figure 6.5: *BR R-graph* for an 8 × 8 BG network.

Figure 6.5. From Figure 6.5, the BR of an $N \times N$ BG network is given by:

$$BR = p_0 \times \left[\prod_{i=1}^{n-1} (1 - (1 - p_i)^2)^{2^i} \right] \times p_{OPC}^{N} \times p_{NMC}, \qquad (6.3)$$

Table 6.5 provides the numerical values for the BR of the BG network using the failure rates provided in Table 6.2. Again, the factors that contribute to the second part of the reliability are not used for the same reason mentioned when we presented the TR calculations for the BG network. The effect of the network size is very obvious on the performance of the BR of the BG network. For a mission period close to three years, the BR deteriorates to 87% for a network of size 128. By inspecting Equation 6.3 we find out that the term p_{OPC}^{N} is responsible for the deterioration of the BR. Table 6.6 depicts the BR levels for the same period in Table 6.5 if we exclude that term (assuming that the all the OPCs are highly reliable). The results in Table 6.6 suggest that the network architecture is not the bottleneck. Instead, it is the reliability of the output stage and represented by the term p_{OPC}^{N} that is significantly affecting the BR

Mission Time (hours ×1000)	8×8	16×16	32×32	64×64	128×128
5	0.9980276	0.9963823	0.9930829	0.9864774	0.9733242
10	0.9960586	0.9927765	0.9862111	0.9731326	0.9473498
15	0.9940930	0.9891825	0.9793844	0.9599633	0.9220588
20	0.9921308	0.9856005	0.9726025	0.9469672	0.8974335
25	0.9901720	0.9820302	0.9658651	0.9341422	0.8734566

Table 6.5: Behavior of the BR for various sizes of the BG network.

Mission Time (hours ×1000)	8×8	16×16	32×32	64×64	128×128
5	0.9996586	0.9996585	0.9996570	0.9996558	0.9996524
10	0.9993168	0.9993159	0.9993117	0.9993065	0.9992944
15	0.9989747	0.9989722	0.9989639	0.9989521	0.9989258
20	0.9986321	0.9986275	0.9986137	0.9985926	0.9985467
25	0.9982892	0.9982818	0.9982611	0.9982280	0.9981572

Table 6.6: Behavior of the BR for various sizes of the BG network by excluding the p_{OPC}^{N} term.

of the BG network. Moreover, this output stage reliability will always limit the BR levels of any other architecture because if one OPC fails the architecture will lose its broadcast property.

One way to improve the BR performance is by increasing the redundancy for the output stage. This can by achieved by using standby units of the OPCs in conjunction with the failure detecting mechanism we discussed in the previous chapter. This might lead to separating the output stage to another chip.

6.4.3 Network Reliability

NR is the probability of maintaining full access property throughout the network. Recall that the BG network will lose full access property *iff* a critical pair fails in the network. The NR of the BG network can be given by:

$$NR = FSR \times \left[\prod_{i=1}^{n-2} SR_i\right] \times LSR \times OSR \times p_{NMC}, \tag{6.4}$$

where

FSR = reliability of the first stage.

SR_i = reliability of $Stage_i$.

LSR = reliability of the $Stage_{n-1}$.

OSR = reliability of the output stage.

and they are given by:

$$FSR = p_0^N, \tag{6.5}$$

$$SR_i = \sum_{j=0}^{N/2} INF_j p_i^{N-j}(1-p_i)^j, \tag{6.6}$$

$$LSR = \sum_{i=0}^{N/2} LNF_i p_{n-1}^{N-1}(1-p_{n-1})^i, \tag{6.7}$$

$$OSR = p_{OPC}^N. \tag{6.8}$$

INF_i and LNF_i indicate all possible combinations of i SE failures in an intermediate stage and the last stage respectively, which do not make the BG network lose full access property. Both INF_i and LNF_i are given by:

$$INF_i = \begin{cases} 1; & i = 0 \\ \binom{N-i}{i} + \binom{N-i-1}{i-1}; & 0 < i \leq N/2. \end{cases} \tag{6.9}$$

$$LNF_i = \sum_{j=0}^{i} \binom{N/2}{j} \times \binom{N/2-j}{i-j}; \qquad 0 \leq i \leq N/2. \tag{6.10}$$

The proofs of Equations 6.9 and 6.10 are provided in Appendix D.

The overall system reliability $(NR_{overall}(t))$ can be given by:

$$NR_{overall}(t) = NR(t) \times e^{-\lambda_{II} t}, \tag{6.11}$$

where the $NR(t)$ is evaluated using Equations 6.4 up to 6.10 and the failure rates provided in Table 6.2 to estimate the reliabilities of the SEs and λ_{II} is obtained from Table 6.2. Figure 6.6 depicts the overall reliability performance for different network sizes for mission periods of up to three years. As expected, smaller networks feature better reliability figures. However, if we exclude both FSR and OSR by assuming both the components of the first stage and the output stage are highly reliable, we obtain very high reliability figures as depicted in Table 6.7. As we discussed previously, the structure of the BG network is not the bottleneck in deciding the overall NR. It is both the FSR and the OSR that decide that performance of the NR. Again, the fact we have just discussed is the same for any other switching architecture. This is because a failure in any of the 1 × 4 SEs or the OPCs will result in losing the full access property of the network.

6.4.4 Mean Time to Failure

The system mean time to failure $(MTTF)$ is given by:

$$MTTF = \int_0^{\infty} NR_{overall}(t)dt. \tag{6.12}$$

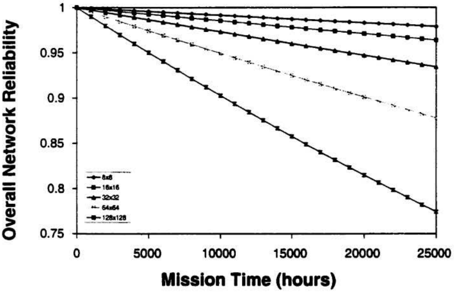


Figure 6.6: ($NR_{overall}$) performance of the BG network.

Mission Time (hours ×1000)	8 × 8	16 × 16	32 × 32	64 × 64	128 × 128
5	0.998719	0.998663	0.998485	0.998188	0.997446
10	0.997439	0.997325	0.996963	0.996355	0.994839
15	0.996159	0.995985	0.995433	0.994501	0.992179
20	0.994881	0.994644	0.993897	0.992626	0.989467
25	0.993602	0.993301	0.992353	0.990730	0.986703

Table 6.7: Behavior of the NR for various sizes of the BG network by excluding both the FSR and OSR terms.

	$lambda_{BG\_I}$	$lambda_{BG\_II}$	$\lambda_{BG\_overall}$
8×8	0.670	0.222	0.892
16×16	1.310	0.233	1.543
32×32	2.564	0.268	2.832
64×64	5.040	0.326	5.3662
128×128	9.957	0.471	10.428

Table 6.8: Estimated BG network failures/10^6 hours.

In general, a system failure rate is given by:

$$\lambda = \frac{1}{MTTF}. \tag{6.13}$$

Table 6.8 provides the failure rate (λ_{BG_I}) for the BG network due the first part of the model as well as the overall cumulative failure rate ($\lambda_{BG\_overall}$). λ_{BG_I} and $\lambda_{BG\_overall}$ are defined as:

$$\lambda_{BG\_I} = \frac{1}{\int_0^\infty NR(t)dt}. \tag{6.14}$$

$$\lambda_{BG\_overall} = \frac{1}{\int_0^\infty NR_{overall}(t)dt}. \tag{6.15}$$

The results in Table 6.8 emphasize the role of the network size in degrading the reliability performance of the network. It is obvious the first part factors of the model are contributing more to the overall system failure rate than the second part factors of the model. However, the contribution of the second part is more significant for smaller networks. For example, the failure rate due to the second part in an 8 × 8 network approximately represents 25% of the overall system failure rate while it only represents 4.5% in 128 × 128 network. Therefore, we can conclude that failure in small networks is partially due to environmental operating conditions, expected number of pins, fabrication quality and packaging. However, failure in large networks is mostly due to density and technology used.

It is interesting to note that the reliability of a 128×128 BG network is better than 0.77 for a three-year mission period. This corresponds to an MTTF of over ten years for the device. Such high reliability figures suggest that these networks are fit for practical implementation. Indeed, when repair is taken into account, very high availability figures can be obtained. As discussed earlier, the current analysis does not include interstage links. However, even when they are included in the reliability model, we expect the overall dependability of the device(s) implementing BG networks to be quite high.

6.5 Summary

The BG network is a single fault-tolerant network and robust in case of multiple faults. In this chapter, we introduced an exact model for the network reliability of the BG network to obtain accurate results. We also adopted a model for failure rate prediction of the very high speed integrated circuits to obtain realistic figures for the BG network reliability metrics. The estimated failure rates are based on 0.8 μm BiCMOS technology, and not the 0.35 μm technology, due to the limitations imposed on the aforementioned model. The model is composed of two parts. The first part represents the technology used and the area of the design. The second part represents the environmental operating conditions, expected number of pins, fabrication quality and packaging. The resulting failure rates are used to evaluate the three reliability metrics terminal reliability, broadcast reliability, and network reliability. The obtained results for these three metrics prove that the BG network is a robust and reliable switch fabric. For example, a 128×128 BG network has a mean time to failure of above 10 years. The results also showed that the deterioration in the BR and NR is mainly due to both the reliability of the first stage and the output stage. In any other switching architecture, both the reliabilities of the first and the output stages will have a similar effect.

Chapter 7

Conclusion and Future Work

The main contributions of this work can be summarized as follows:

- **Demonstrating the outstanding performance of the BG network.** The performance of the BG network has proven its firm competitiveness with other switching architectures. A single plane BG (BG-1) configuration has shown to be an outstanding switching architecture under various types of traffic loads. Under a wide range of URT, bursty, and nonuniform traffic loads, we discovered two important facts. Firstly, the performance of BG-1 configuration is almost the same as the performance of the hypothetical ideal network under the above mentioned loads. Secondly, we discovered that the performance of a single plane crossbar (crossbar-1) configuration lags far behind the ideal and the BG-1 configurations. The results provided in Chapter 4 demonstrate these two facts. These results are obtained under realistic scenarios of finite buffering budgets rather than infinite ones to capture the effect of the queuing problems in these architectures. By scanning the results of Chapter 4 we can conclude that it is not recommended that switching architectures be operated under heavy loads. This is because the ideal network, which provides the best upper bound for any performance parameter, has shown very high levels of average and maximum cell delays under bursty traffic. As we know, B-ISDN traffic is going to be bursty in nature. As we discussed in Chapter 4, the efficient performance of the BG network is due to its architectural feature that enables receiving up to 4 cells by the OPCs in any switching cycle.

- **New routing algorithm**. In this dissertation, we introduced a new routing algorithm for the BG network which is as simple as the routing algorithm used in the banyan network. As we showed in Chapter 3, the new algorithm necessitated modifications to the earlier proposed topologies of the BG network. These new modifications did not subdue the low complexity or the high modularity enjoyed by the BG network. The new algorithm has reduced the routing decision in any SE to only one bit for each arriving cell. This has enables us to comfortably introduce two levels of priority for the cells handled by the new topology.

- **Realizable architecture**. One of the major tasks in this work is to show that the BG network is not only an attractive network, as far as the performance analysis is concerned, but also to prove that it is realizable as far as the hardware complexity is concerned. In Chapter 5, we introduced a comprehensive front-end design of the BG network. The modular and scalable architecture of the BG network has facilitated a hierarchical bottom-up and smooth design process. The modularity has directed us to firstly design the building blocks – the switching element (SE), the output port controller (OPC), and the network main controller (NMC) – and then use them to build the higher level modules in the system hierarchy. The scalability of the BG network has enabled us to parameterize the RTL description of the design. By changing a few parameters in one of the header files we can obtain a new and different realization of the BG network. These parameters are the network size N, the buffer size in the OPCs, and the cell size. Due to the strong requirement for testability in VLSI systems, we included a BIST feature in the design. We also made use of the BIST feature to build a self-diagnostic and repair mechanism which takes advantage of the fault tolerance properties of the BG network. The self-diagnostic and repair mechanism can localize faults and take the proper decisions to tolerate these faults. The results we obtained for our design suggested that the network can operate comfortably at a speed of 200 MHz. These results are based on a 0.35 μm CMOS technology.

- **Accurate reliability modelling.** Designing highly reliable systems is a crucial requirement in the industry of broadband communications where consequences of the system failures are very expensive. Accordingly, we introduced an exact modelling of the BG network reliability. Moreover, to assess the network reliability performance we adopted a model for failure rate prediction of the very high speed integrated circuits to obtain realistic figures for the BG network reliability metrics. This adopted model is based on a 0.8 μm BiCMOS technology. The adopted model is composed of two parts. The first part represents the technology used and the area of the design. The second part represents the environmental operating conditions, expected number of pins, fabrication quality and packaging. The resulting failure rates are used to evaluate the three reliability metrics: terminal reliability, broadcast reliability, and network reliability. The results obtained in Chapter 6 for these three metrics proved that the BG network is a robust and reliable switch fabric. For example, a 128×128 BG network has a mean time to failure of above 10 years. The results also showed that the deterioration in the reliability metrics is mainly due to both the reliability of the first stage and the output stage. In any other switching architecture, both the reliabilities of the first and the output stages will have a similar effect.

Accordingly, we can conclude the following:

- The BG architecture is a very strong candidate to be used in broadband communication switch fabrics. The performance under wide range of loads of various traffic models and the realistic reliability analysis of the BG architecture have proven this strong candidacy. This candidacy is also affirmed by the easy and uniform VLSI design of the architecture.

- This study challenges the strong belief by the broadband communication industry that the "nonblocking" crossbar architecture is the best candidate for broadband switch fabrics fabrics. The crossbar configurations have shown less efficient performance under the various traffic models when compared to the BG configurations. In addition, the crossbar has more complicated VLSI design process due to its nonuniform architecture.

- It is not recommended that switch fabrics run under heavy loads all the time because this leads to undesirable high levels of delay and cell loss. This conclusion is based on the performance results obtained for the ideal network performance and various traffic loads. This is true for all existing switch fabrics and also the current practice.

7.1 Future Work

Although the BG network has shown outstanding performance under different types of traffic loads, we still need to investigate the performance under other scenarios. Scheduling is a very important issue as far as the quality of service is concerned in broadband communication networks. The term scheduling refers to the mechanism that determines what queue is given an opportunity to transmit. A queuing structure and a scheduling mechanism attempt to achieve [94] flexibility, scalability, efficiency, guaranteed QoS, isolation, and fairness. IPCs and OPCs are the locations where scheduling mechanisms are needed to be investigated in the BG network. The current structure of the first stage can only support one single queue at the first stage. One modification that facilitates supporting two queues can be achieved by replacing the 1×4 SEs in the first stage by 2×4. Another modification which supports four queues in each IPC can be also achieved by building the first stage out of 4×4 SEs. That is, the whole BG network will be based on 4×4 SEs. The hardware complexity and the realizability of the BG network will need to be investigated to know the impact of adding new service priorities.

Buffer sharing is a well-known method, where the buffering resources of adjacent IPCs or OPCs are shared. As we discussed in Chapter 4, buffer sharing results in the lowest buffering requirements. However, this is achieved at the expense of more complex buffer control mechanisms, especially when the level of sharing increases. Accordingly, the performance and the implementation issues should be studied to investigate the cost-effectiveness of buffer sharing.

Another issue that needs to be addressed is the performance analysis under multicast traffic loads. Multicasting has always attracted the attention of researchers and designers. In [95], an extensive survey of the multicast switches for the period between 1984 and 1997 is presented with a timetable showing the history of multicast switches. The authors have reached the conclusion that there are three families of multicast switches: 1) multicast switches without copy networks, 2) multicast switches with copy networks without prescheduling, 3) prescheduling before copying. The survey conducted was only for space division switches and it was discovered that multicast switches were based on one of the following architectures: 1) knockout switch, 2) banyan network, 3) Clos network. In fact, multicasting is a complicated problem and all the proposed architectures cannot guarantee a complete solution for it. Both the design and performance issues should be investigated for any multicasting method proposed for the BG network.

Finally, it would be optimum to test performance of the BG network under real B-ISDN traffic loads. However, it seems this is not currently feasible because the real B-ISDN traffic is not established yet. Perhaps, testing the performance under mixes of different bursty sources would be a more realistic situation. However, this will lead to longer simulation periods as the amount of computations needed for traffic generation will escalate.

References

[1] O. Kyas, *ATM Networks*. International Thomson Computer Press, 1996.

[2] H. Yamanaka *et al.*, "Scalable Shared-Buffering ATM Switch with a Versatile Searchable Queue," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 773–784, June 1997.

[3] D. Weil *et al.*, "A 16 × 622 Mb/s ATM Switch: PRELUDE Switch Architecture Integrated into a 6-Million Transistor Monochip," *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 1108–1114, July 1997.

[4] S. Kothari, G. Prabhu, and R. Roberts, "The Kappa Network with Fault-Tolerant Destination Tag Algorithm," *IEEE Transactions on Computers*, vol. 37, pp. 612–617, May 1988.

[5] R. Y. Awdeh and H. Mouftah, "Survey of ATM Switch Architectures," *Computer Networks and ISDN Systems*, vol. 27, pp. 1567–1613, November 1995.

[6] D. Basak, A. K. Choudhury, and E. L. Hahne, "Sharing Memory in Banyan-Based ATM Switches," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 881–891, June 1997.

[7] S. Q. Li, "Nonuniform Traffic Analysis on a Nonblocking Space-Division Packet Switch," *IEEE Transactions on Communication*, vol. 38, pp. 1085–1096, July 1990.

[8] M. Abramovici, M. A.Breuer, and A. D.Friedman, *Digital Systems Testing and Testable Design.* Computer Science Press, 1990.

[9] Rod Byrne, *A High-Level Language and CAD Enviornment for BIST Embedding.* PhD thesis, University of Victoria, 1994.

[10] P. Wong and M. Yeung, "Design and Analysis of a Novel Fast Packet Switch–Pipeline Banyan," *IEEE Transactions on Networking*, vol. 3, pp. 63–69, February 1995.

[11] M. Bentall, C. Hobbs, and B. Turton, *ATM and Internet Protocol: A Convergence of Technologies.* Arnold Inc., 1998.

[12] W. Stallings, *ISDN and Broadband ISDN with Frame Relay and ATM.* Prentice Hall, Fourth Edition, 1999.

[13] S. Keshav, *An Engineering Approach to Computer Networds: ATM Networks, the Internet, and the Telephone Networks.* Addison Wesly, 1997.

[14] *http://www.atmforum.com/.*

[15] W. R. Stevens, *TCP/IP Illustrated, Volume 1.* Addison-Wesley, 1994.

[16] M. Guizani and A. Rayes, *Designing ATM Switching Networks.* McGraw Hill, 1999.

[17] E. R. Coover, *ATM Switches.* Artech House Inc., 1997.

[18] P. Newman, "ATM Technology for Corporate Networks," *IEEE Communication*, vol. 30, pp. 90–101, April 1992.

[19] F. A. Tabagi, "Fast Packet Switch Architectures for Broadband Integrated Services Digital Networks," *Proceedings of the IEEE*, vol. 78, pp. 133–167, January 1990.

[20] J. Turner and N. Yamanaka, "Architectural Choices in Large Scale ATM Switches," *IEICE Transactions on Communications*, vol. E81-B, pp. 120–137, February 1998.

[21] H. Ahmadi and W. E. Denzel, "A Survey of Modern High-Performance Switching Techniques," *IEEE Journal on Selected Areas in Communications*, vol. 7, pp. 1091–1103, September 1989.

[22] E. W. Zegura, "Architectures for ATM Switching Systems," *IEEE Communication Magazine*, vol. 31, pp. 28–37, February 1993.

[23] R. Rooholamini, V. Cherkassky, and M. Garver, "Finding the Right ATM Switch for the Market," *IEEE Computer*, vol. 27, pp. 16–28, April 1994.

[24] L. T. Lee and P. H. Huang, "A High Speed Factorial Style Memory Switch Architecture," *IEICE Transactions on Communications*, vol. E81-B, pp. 164–174, February 1998.

[25] A. Thomas, J. Coudreuse, and M. Servel, "Asynchronous Time-division Switching: An Experimental Packet Network integrating Video-communicatins," in *Proceedings of ISS'84*, p. paper 32C2, May 1984.

[26] A. Chemarin *et al.*, "A High-speed CMOS circuit for 1.2-Gbits/s 16 × 16 ATM Switching," *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 1116–1120, July 1992.

[27] K. Eng and M. Pashan, "Advances in Shared-Memory Designs for Gigabit ATM Switching," *Bell Labs Technical Journal*, pp. 175–187, Spring 1997.

[28] M. Praks, "A modular Element for Shared Buffer ATM Switch Fabric," in *Proceedings of the 1997 IEEE International Conference on Application Specific Systems, Architecture, and Processors*, pp. 432–436, 1997.

[29] T. K. Woo, "Design and Performance Analysis of Crossbar ATM Switching Architecture," *Computer Communications*, vol. 21, pp. 88–94, 1998.

[30] M. Hluchyj and M. Karol, "Queuing in High-performance Packet Switching," *IEEE Journal on Selected Area in Communications*, vol. 6, pp. 1587–1597, December 1988.

[31] M. Karol *et al.*, "Input Versus Output Queuing in A Space Division Packet Switch," *IEEE Transactions on Communications*, vol. COM-35, pp. 1347–1356, December 1987.

[32] J. Hui and E. Arthurs, "A Broadband Packet Switch for Integrated Transport," *IEEE Journal on Selected Areas in Communications*, vol. SAC-5, pp. 1264–1273, October 1987.

[33] N. Mckeown *et al.*, "Tiny Tera: A Packet Switch Core," *IEEE Micro*, pp. 26–33, January/February 1997.

[34] K. Genda, N. Yamanaka, and Y. Doi, "A 160 GB/s ATM Switch Using Internal Speed-up Crossbar Switch Architecture," *Electronics and Communications in Japan*, vol. 80, pp. 68–78, September 1997.

[35] C. Goke and G. Lipovski, "Banyan Networks for Partitioning Multiprocessor Systems," in *First Annual Symposium on Computer Architecture*, pp. 21–28, 1973.

[36] J. H. Patel, "Peformance of Processor-Memory Interconnections for Multiprocessors," *IEEE Transactions on Computers*, vol. C-30, pp. 771–780, October 1981.

[37] C. Wu and Y. Feng, "On a Class of Multistage Interconnection Networks," *IEEE Transactions on Computers*, vol. 29, pp. 694–702, August 1980.

[38] D. Lawrie, "Access and Alignment of Data in an Array Processor," *IEEE Transactions on Computers*, vol. 24, pp. 1145–1155, December 1975.

[39] M. Pease, "The Indirect Binary n-Cube Multiprocessor Array," *IEEE Transactions on Computers*, vol. 26, pp. 458–473, May 1977.

[40] D. Dias and M. Kumar, "Packet Switching in $N\log N$ Multistage Networks," in *Proceedings of IEEE GLOBECOM'84*, pp. 114–120, 1984.

[41] C. Clos, "A Study of Nonblocking Switching Networks," *IEEE Transactions on Computers*, pp. 406–424, March 1953.

[42] J. Beetem, M. Denneau, and D. Weingarten, "The GF-11 Supercomputer," in *Proceedings of the* 12^{th} *Annual Symposium on Computer Architecture*, pp. 108–115, 1985.

[43] V. Beneš, *Mathematical Theory of Connecting Networks and Telephone Traffic*. Academic Press, 1965.

[44] S. Ohta, "A Simple Control Algorithm for Rearrangeable Switching Networks with Time Division Multiplexing Links," *IEEE Journal on Selected Areas in Communications*, vol. 5, pp. 1302–1308, October 1987.

[45] F. F. Liotopoulos and S. Chalasani, "Semi-Rearrangeably Nonblocking Operation of Clos Networks in the Multirate Environment," *IEEE Transactions on Networking*, vol. 4, pp. 281–291, April 1996.

[46] S. Sabesan, W. A. Crossland, and R. W. Scarr, "Nonblocking ATM Switching Networks Composed of ATM Switching Modules," in *Proceedings of IEEE GLOBECOM'97*, pp. 232–236, 1997.

[47] R. Melen and J. Turner, "Nonblocking Multirate Distribution Networks," *IEEE Transactions on Communications*, vol. 41, pp. 362–369, February 1993.

[48] K. Eng, M. Karol, and Y. Yeh, "A Growable Packet (ATM) Switch Architecture: Design Principles and Applications," in *Proceedings of IEEE GLOBLECOM'89*, pp. 1159–1165, 1989.

[49] T. T. Lee and C. H. Lam, "Path Switching–A Quasi-Static Routing Scheme for Large-Scale ATM Packet Switches," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 914–924, June 1997.

[50] Naoaki *et al.*, "OPTIMA: Tb/s ATM Switching System Architecture," in *Proceedings of the IEEE ATM'97 Workshop*, pp. 691–696, 1997.

[51] D. Parker and C. Raghavendra, "The Gamma Network," *IEEE Transactions on Computers*, vol. 33, pp. 367–373, April 1984.

[52] J. Hui, *Switching and Traffic Theory for Integrated Broadband Networks.* Kluwer Academic Publishers, Boston, 1990.

[53] K. Batcher, "Sorting Networks and Their Applications," in *Proceedings of AFIPS Spring Joint Computer Conference*, pp. 307–314, 1968.

[54] H. Sivakumar and R. Venkatesan, "Blocking in Multistage Interconnection Networks for Broadband Packet Switch Architectures," in *Proceedings of the Sixth Annual Newfoundland Electrical and Computer Engineering Conference*, 1995.

[55] B. C. Lindberg, *Digital Broadband Networks & Services*. McGraw-Hill Series on Computer Communications, 1994.

[56] M. D. Marco and A. Pattavina, "Distributed Routing Protocols for ATM Extended Banyan Networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 925–937, June 1997.

[57] J. Turner, "Design of a Broadcast Packet Switching Network," *IEEE Transactions on Communications*, vol. 36, pp. 734–743, June 1988.

[58] C. Kruskal and M. Snir, "The Performance of Multistage Interconnection Networks for Multiprocessors," *IEEE Transactions on Computers*, vol. c-32, pp. 1091–1098, December 1983.

[59] M. Alimuddin, H. Alnuweiri, and R. Donaldson, "The Fat Banyan ATM Switch," in *Proceedings of the IEEE INFOCOM'95*, vol. 2, pp. 659–666, 1995.

[60] F. A. Tobagi, T. Kwok, and F. M. Chiussi, "Architecture, Performance, and Implementation of the Tandem Banyan Fast Packet Switch," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1173–1193, October 1991.

[61] J. Turner, "New Directions in Communicarions (or Which Way to The Information Age?)," *IEEE Communications Magazine*, pp. 8–15, October 1986.

[62] P. Wong and M. Yeung, "Pipeline Banyan - A Parallel Fast Packet Switch Architecture," in *Proceedings of the ICC'92*, pp. 882–887, 1992.

[63] T. Cheng and Y. Shen, "Performance Analysis of Parallel Banyan ATM Switch," *International Journal of Communications*, vol. 10, pp. 43–52, 1997.

[64] R. Venkatesan and H. Mouftah, "Balanced Gamma Network - A New Candidate for Broadband Packet Switching Architectures," in *Proceedings of the IEEE INFOCOM'92*, vol. 3, pp. 2482–2488, 1992.

[65] Harinath Sivakumar, "Performance, Fault Tolerance and Reliability of Multistage Interconnection Netwroks for Broadband Packet Switch Architectures," Master's thesis, Memorial University of Newfoundland, 1995.

[66] P. Goli and V. Kumar, "Performance of a crosspoint Buffered ATM Switch Fabric," in *Proceedings of the IEEE INFOCOM'92*, vol. 1, pp. 426–435, 1992.

[67] B. Zhou and M. Atiquzzaman, "Efficeint Analysis of Multistage Interconnection Networks Using Finite Output-Buffered Switching Elements," *Computer Networks and ISDN Systems*, vol. 28, pp. 1809–1829, 1996.

[68] V.P. Kumar *et al.*, "PHOENIX: A Building Block for Fault Tolerant Broadband Packet Switches," in *Proceedings of the IEEE GLOBECOM'91*, pp. 228–233, 1991.

[69] W. W. Hines and D. C. Montgomery, *Probability and Statistics in Engineering and Management Science.* John Wiley & Sons, 1980.

[70] Y. E. Sayed and R. Venkatesan, "Modeling and Simulation of the Pipelined Balanced Gamma Network," in *Proceeding of the Fourth IEEE International Conference on Electronics, Circuits, & Systems (ICECS'97)*, vol. 1, pp. 97–101, 1997.

[71] A. Pattavina, *Switching Theory: Architecture and Performance in Broadband ATM Networks.* Wiley, 1998.

[72] H. F. Badran and H. T. Mouftah, "Performance of Broadband Integrated Switch Architectures with Input-Output-Buffering under Backpressure Mechanisms." Tech. Rep. 90-7, Queen's University, Kingston, Ontario, Canada, 1990.

[73] G. D. Stamoulis, M. Anagnostou, and A. Georgantas, "Traffic Source Models for ATM Networks: A Survey," *Computer Communications*, vol. 17, no. 6, pp. 428–438, 1994.

[74] J. Banks, J. S. Carson, and B. L. Nelson, *Discrete-Event System Simulation.* Second Edition, Prentice Hall, 1996.

[75] Royal Military College of Canada and Canadian Microelectronics Corporation, *Instruction on Basic Digital IC Design Flow From RTL Description to Completed CMOS Design Using Cadence (97A) and Synopsys*, November 1998. Document ICI-089.

[76] N. Mukherjee, T. J.Chakraborty, and R. Karri, "Built-in Self-Test: A complete Test Solution for Telecommunication Systems," *IEEE Communications Magazine*, vol. 37, pp. 72–78, June 1999.

[77] J. Turino, *Design to Test.* Van Nostrand Reinhold, 1990.

[78] G. Russel and I. L. Sayers, *Advanced Simulation and Test Methodologies for VLSI design.* Van Nostrand Reinhold, 1989.

[79] M. H. Guo and R. S. Chang, "Mulitcast ATM Switches: Survey and Performance Evaluation," *ACM SIGCOMM Computer Communication Review*, vol. 28, pp. 98–131, April 1998.

[80] F. J. Hill and G. R. Peterson, *Computer Aided Logical Design with Emphasis on VLSI.* John Wiley & Sons, 1993.

[81] http://www-cad.eecs.berkeley.edu:80/Software/software.html.

[82] G. Adams, D. Agrawal, and H. Siegel, "A Survey and Comparison of Fault-Tolerant Multistage Interconnection Networks," *IEEE Computer*, vol. 20, pp. 14–27, June 1987.

[83] V. Cherkassky and M. Malek, "Reliability and Fail-Softness Analysis of Multistage Interconnection Networks," *IEEE Transactions on Reliability*, vol. 34, pp. 524–527, December 1985.

[84] J. P. Provan, "Bounds on the Reliability of Networks," *IEEE Transactions on Reliability*, vol. 35, pp. 260–268, August 1986.

[85] J. Blake and K. Trivedi, "Reliability Analysis of Interconnection Networks Using Hierarchical Composition," *IEEE Transactions on Reliability*, vol. 38, pp. 111–119, April 1989.

[86] A. Varma and C. Raghavendra, "Reliability Analysis of Redundant-Path Interconnection Networks," *IEEE Transactions on Reliability*, vol. 38, pp. 130–137, April 1989.

[87] C. Botting, S. Rai, and D. Agrawal, "Reliability Computation of Multistage Interconnection Networks," *IEEE Transactions on Reliability*, vol. 38, pp. 138–145, April 1989.

[88] J. Blake and K. Trivedi, "Multistage Interconnection Network Reliability," *IEEE Transactions on Reliability*, vol. 38, pp. 1600–1604, November 1989.

[89] P. Tagle and N. Sharma, "A High Performance Fault-Tolerant Switching Network for B-ISDN," in *Proceedings of the IEEE 14th Annual Intl. Phoenix Conf. on Computer And Communications*, pp. 599–606, 1995.

[90] P. Tagle and N. Sharma, "Performance of Fault-Tolerant ATM Switches," *IEE Proceedings on Communication*, vol. 143, no. 5, pp. 317–324, 1996.

[91] Y. E. Sayed, R. Venkatesan, and H. Sivakumar, "Fault Tolerance and Reliability Analysis of the Balanced Gamma Network," *International Journal of Parallel and Distributed Systems and Networks*, vol. 2, no. 4, pp. 244–254, 1999.

[92] *Military Handbook, Reliability Prediction of Electronic Equipment.* MIL-HDBK-217F, 1991, (Updated in Feb. 1995).

[93] D. Agrawal and J. Leu, "Dynamic Accessibility Testing and Path Length Optimization of Multistage Interconnection Networks," *IEEE Transactions on Computers*, vol. c-34, pp. 255–266, March 1985.

[94] N. Giroux and S. Ganti, *Quality of Service in ATM Networks: State-of-the-Art Traffic Managment.* Prentice Hall PTR, 1999.

[95] M. Guo and R. Chang, "Multicast ATM Switches: Survey and Performance Evaluation," *SIGCOMM Computer Communication Review*, vol. 28, pp. 98–131, April 1998.

Appendix A

Balanced Gamma Network Topology

$IL_i \equiv$ input link i.

$OL_i \equiv$ output link i.

```
for ($j = 0; j < n\ ; j$++)                // $j$ is the stage index
    for ($i = 0; i < N; i$++)              // $i$ is the row index
        $\alpha = \lfloor \frac{i}{2^j} \rfloor \bmod 2$.
            if ($\alpha = 0$)
                connect $OL_0$ to $IL_0$ of $SE_{i,j+1}$.
                connect $OL_1$ to $IL_1$ of $SE_{i+2^{j+1},j+1}$.
                connect $OL_2$ to $IL_2$ of $SE_{i-2^j,j+1}$.
                connect $OL_3$ to $IL_3$ of $SE_{i+2^j,j+1}$.
            else
                connect $OL_0$ to $IL_2$ of $SE_{i-2^j,j+1}$.
                connect $OL_1$ to $IL_3$ of $SE_{i+2^j,j+1}$.
                connect $OL_2$ to $IL_0$ of $SE_{i,j+1}$.
                connect $OL_3$ to $IL_1$ of $SE_{i+2^{j+1},j+1}$.
```

Appendix B

Balanced Gamma Network Routing Algorithm

The following notation is important to understand the following pseudo code:

```
output[i]= holds which input is connected to output 'i'.

priority[i]= holds the priority of cell at input 'i',
           = -1      no cell exists.
           = 0       low priority cell.
           = 1       high priority cell.

state_up = number of cells destining up.
state_down = number of cells destining down.

cell_status = status of cell whether going up or down.

ack[i] = holds the acknowledgement for cell arriving at input 'i',
       = 0       cell is dropped.
       = 1       cell is accepted.
```

```
output_decision

   output[0]=output[1]=output[2]=output[3]=4; // just a value out of
                                              // range to start with
   if( (state_up<=2) && state_down <=2) // In that case no conflict
                                        // will take place.

       if(priority[1]>-1) // active cell at input '1'.

           if(cell_status[1]==up)
               output[0]=1;
           else
               output[2]=1;
```

```
if (priority[3]>-1)  // active cell at input 2.

    if (cell_status[3] == up) // cell destining up

        if(priority[output[0]]<=priority[3])
            output[1]=output[0];
            output[0]=3;
        else
            output[1]=3;

    else // cell destining down

        if(priority[output[2]]< priority[3])
            output[3]=output[2];
            output[2]=3;
        else
            output[3]=3;


if (priority[2] > -1) // input[2] is active

    if(cell_status[2]==up)

         if(priority[output[0]]<=priority[2])
            output[1]=output[0];
            output[0]=2;
        else
            output[1]=2;

    else

        if(priority[output[2]]<=priority[2])
            output[3]=output[2];
            output[2]=2;
        else
            output[3]=2;


if( priority[0]>-1)  // active cell at input '2'.

    if(cell_status[0]==up)

        if(priority[output[0]]< priority[0])
              output[1]=output[0];
              output[0]=0;
        else
              output[1]=0;

    else  // cell is destining down.
```

```
            if(priority[output[2]]<priority[0])
                output[3]=output[2];
                output[2]=0;
            else
                output[3]=0;


if(state_up>2)  // more than two cells destining up.

    int count_up_now;  // an internal counter of how many cells
                       // has granted its request to be routed up.
    count_up_now=0;
    if(priority[3]>-1) // a cell is arriving at input '3'.

        if(cell_status[3]==up)
            output[0]=3;
            count_up_now++;
        else // cell is destining down.
            output[2]=3;

    if(priority[1]>-1)  // a cell is arriving at input '1'.

        if(cell_status[1]==up)

            if(priority[output[0]]<priority[1])
                output[1]=output[0];
                output[0]=1;
            else
                output[1]=1;

count_up_now++;

        else  // a cell is destining down.
            output[2]=1;

    if(priority[2]>-1)  // a cell is arriving at input '2'.

        if(cell_status[2]==up)

            switch (count_up_now)

                case 2: // two cells has been already
                        // granted there request to berouted up
                    if(priority[2]>=priority[output[0]])

                        if(priority[output[0]]>priority[output[1]])
                            output[1]=output[0];

            output[0]=2;
                    else
```

```
                        if(priority[2]>=priority[output[1]])
                            output[1]=2;
                    break;

    case 1:
                    if(priority[output[0]]<=priority[2])
                        output[1]=output[0];
                        output[0]=2;
                    else
                        output[1]=2;

                    count_up_now++;
                    break;

case 0:
                    output[0]=2;
                    break;


        else // cell is destining down
           output[2]=2;


    if(priority[0]>-1) // a cell is arriving at input '0'.

        if(cell_status[0]==up)

            switch (count_up_now)

                 case 2:
                    if(priority[0]>=priority[output[0]])

                       if(priority[output[0]]>=priority[output[1]])
                           output[1]=output[0];

                       output[0]=0;
                    else
                       if(priority[0]>priority[output[1]])
                           output[1]=0;
                    break;

                case 1:
                    if(priority[output[0]]<=priority[0])
                        output[1]=output[0];
                        output[0]=0;
                    else
                        output[1]=0;

                    count_up_now++;
                    break;
                case 0:
                    output[0]=0;
                    break;


        else  // a cell is destining down
            output[2]=0;



if (state_down >2) // It is the same thing we did above but
                   // different in that there more than two
                   // cells destining down.

    int count_down_now;  // same as count_up_now discussed above.
    count_down_now=0;
    if(priority[1]>-1)  // cell is arriving at input '1'.

        if(cell_status[1]==down)
             count_down_now++;
             output[2]=1;
        else
             output[0]=1; // cell destining up.

    if(priority[3]>-1)  // cell is arriving at input '3'.

        if(cell_status[3]==down)

            count_down_now++;
            if(priority[output[2]]<=priority[3])
                output[3]=output[2];
                output[2]=3;
            else
                output[3]=3;

        else // cell destining up.
            output[0]=3;

    if(priority[0]>-1)  // a cell is arriving at input '0'.

    if(cell_status[0]==down)

            switch (count_down_now)

                case 2:
                    if(priority[0]>=priority[output[2]])
                        if(priority[output[2]]>priority[output[3]])
                            output[3]=output[2];

                        output[2]=0;
                    else
                    if(priority[0]>=priority[output[3]])
                        output[3]=0;
                break;

            case 1:
                if(priority[output[2]]<=priority[0])
                    output[3]=output[2];
                    output[2]=0;
                else
                    output[3]=0;

                count_down_now++;
                break;

            case 0:
                output[2]=0;
                break;

    else  // cell destining up.
        output[0]=0;

if(priority[2]>-1)

    if(cell_status[2]==down)

        switch (count_down_now)

            case 2:
                if(priority[2]>=priority[output[2]])

                    if(priority[output[2]]>priority[output[3]])
                        output[3]=output[2];

                    output[2]=2;
                else
                    if(priority[2]>=priority[output[3]])
                        output[3]=2;
                break;

             case 1:
                if(priority[output[2]]<=priority[2])
                    output[3]=output[2];
                    output[2]=2;
                else
                    output[3]=2;

                count_down_now++;
                break;

             case 0:
                output[2]=2;
```

```
                       break;

           else
               output[0]=2;

     // end of routine output_decision


ack_decision

  ack[0]=ack[1]=ack[2]=ack[3]=-1; // just an initialization.
   for(int i=0; i<4;i++) // acknowledging accepted cells by '1'

       if(priority[output[i]]>-1) // acknowledge received cells
           ack[output[i]]=1;

   for(int j=0;j<4;j++)// connecting idle outputs to idle or
                       // dropped inputs

       if(priority[output[j]]<0)  // if output 'j' is not
                                  // used at all, scan one of
                                  // the idle inputs and connect
                                  // this output to the first
                                  // idle input found.

           if(ack[j]<0)
               output[j]=j;
           else
               if(ack[(j+2) % 4] < 0)
                   output[j]= (j+2) % 4;
               else
                   if(ack[(j+1) % 4] < 0)
                       output[j] = (j+1) % 4;
                   else
                       if(ack[(j+3) % 4])
                           output[j]= (j+3) % 4;

   for(int k=0; k<4; k++)// acknowledging dropped cells by '0'

       if(priority[k]>-1&&ack[k]<0)
           ack[k]=0;
```

Appendix C

Thoughput Under Uniform Random Traffic

In [64], a recursive system of equations for the TP of the BG network was achieved. The analysis assumed only one type of cell priority. Both OL_0 or OL_2 are called the output regular links; because they are the favoured output links in the case of only one cell having a routing bit value of 0 or 1, respectively. Both OL_1 or OL_3 are called output alternate links because they are used if more than one cell has the same routing bit. The probability that a cell will get routed through an output regular link in an SE located at stage i is denoted by $x_{r,i}(1)$. The probability that no cell will get routed through an output regular link in an SE located at stage i is denoted by $x_{r,i}(0)$. Similary, the probability a cell will (will not) get routed through an output alternate link of an SE located at stage i is denoted by $x_{a,i}(1)$ $(x_{a,i}(0))$.

In $stage_0$, for any regular link we can write $x_{r,0}(0) = x_{r,0}(1) = \rho/2$, and for any alternate link we can write $x_{a,0}(0) = x_{a,0}(1) = 0$. We can also write the following recursive equations:

$$\begin{aligned} x_{r,i}(0) &= x^2_{r,i-1}(0)x^2_{a,i-1}(0) + x^2_{r,i-1}(0)x_{a,i-1}(0)x_{a,i-1}(1) \\ &+ x_{r,i-1}(0)x_{r,i-1}(1)x^2_{a,i-1}(0) + \frac{1}{4}x^2_{r,i-1}(1)x^2_{a,i-1}(0) \\ &+ \frac{1}{4}x^2_{r,i-1}(0)x^2_{a,i-1}(1) + x_{r,i-1}(0)x_{r,i-1}(1)x_{a,i-1}(0)x_{a,i-1}(1) \end{aligned}$$

$$\begin{aligned} &+ \frac{1}{4}x^2_{r,i-1}(1)x_{a,i-1}(0)x_{a,i-1}(1) + \frac{1}{4}x_{r,i-1}(0)x_{r,i-1}(1)x^2_{a,i-1}(1) \\ &+ \frac{1}{16}x^2_{r,i-1}(1)x^2_{a,i-1}(1), \end{aligned} \tag{C.1}$$

$$x_{r,i}(1) = 1 - x_{r,i}(0), \tag{C.2}$$

$$\begin{aligned} x_{a,i}(1) &= \frac{1}{4}x^2_{r,i-1}(1)x^2_{a,i-1}(0) + \frac{1}{4}x^2_{r,i-1}(0)x^2_{a,i-1}(1) \\ &+ x_{r,i-1}(0)x_{r,i-1}(1)x_{a,i-1}(0)x_{a,i-1}(1) + x^2_{r,i-1}(1)x_{a,i-1}(0)x_{a,i-1}(1) \\ &+ x_{r,i-1}(0)x_{r,i-1}(1)x^2_{a,i-1}(1) + \frac{11}{16}x^2_{r,i-1}(1)x^2_{a,i-1}(1), \end{aligned} \tag{C.3}$$

$$x_{a,i}(0) = 1 - x_{a,i}(1). \tag{C.4}$$

With four cells accepted by each output port, the TP of the BG network is given by:

$$TP = 2x_{r,i}(1) + 2x_{a,i}(1). \tag{C.5}$$

Appendix D

Calculation of INF_i and LNF_i

Calculations of INF_i

Without loss of generality, we assume that the N critical pairs in any of the intermediate stages are overlapped as shown in Figure D.1.a. The modulo N relationship can be depicted by a circle as shown in Figure D.1.b. Now we want to evaluate the total number of combinations in the intermediate stage such that the network does not lose full access property. Clearly, this is the total number of combinations such that the faulty SEs do not constitute critical pairs. Moreover, the problem can be formulated in the following combinatorial form:

> *If we have N distinctive sites located consecutively to each other on a circle and each site is capable of holding at most one binary number, 0 or 1, what is the total number of combinations if we place i $(i \leq N/2)$ identical 1s in these sites under the condition that no two 1s can reside in two consecutive sites? i.e. there should be a separation by at least one 0 between any two 1s.*

This problem can be divided into two other problems as shown in Figure D.2 by unfolding the circle problem into two queue problems. The first problem illustrated in Figure D.2.a, is to find the total number of combinations of putting i 1s in a queue of

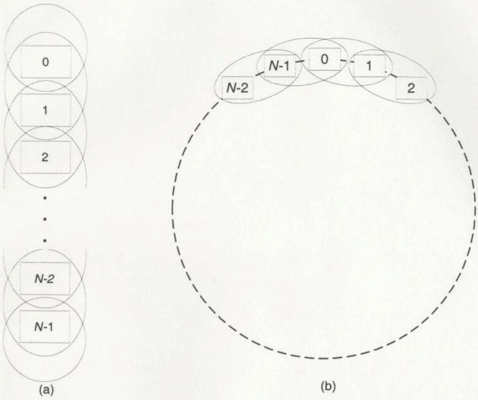


Figure D.1: The model of the critical pairs for an intermediate stage.

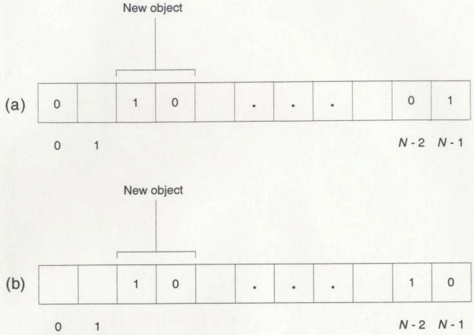


Figure D.2: Expansion of the problem into two queues.

N sites such that the last site contains 1 and any two adjacent sites must not contain 1s. The second problem illustrated in Figure D.2.b, is to find the total number of combinations of putting i 1s in a queue of N sites such that the last site contains 0 and any two adjacent sites must not contain 1s.

To solve the first problem, we reformulate the problem by associating a 0 with each 1 and excluding the last site with the 1 residing in it. If we assume the associated 1 and 0 constitute an object, then the problem reduces to finding the total number of combinations of arranging $i-1$ objects with $(N-2)-(2i-2)(=N-2i)$ zeros. This is easily given by $\begin{pmatrix} N-i-1 \\ i-1 \end{pmatrix}$. Similarly, associating a 0 with each 1 solves the second problem. The total number of combinations in this case is $\begin{pmatrix} N-i \\ i \end{pmatrix}$. Finally, INF_i is the sum of the above two values.

Calculation of LNF_i

There are $N/2$ critical pairs in $Stage_{n-1}$ as mentioned in Section 6.3. We can deal with the problem easily by dividing the SEs into two groups such that any group does not contain any critical pairs as shown in Figure D.3. LNF_i can be given by:

$$LNF_i = \sum_{j=0}^{i} \begin{pmatrix} N/2 \\ j \end{pmatrix} \times \begin{pmatrix} N/2-j \\ i-j \end{pmatrix}; \qquad 0 \leq i \leq N/2, \qquad \text{(D.1)}$$

where $\begin{pmatrix} N/2 \\ j \end{pmatrix}$ represents the total number of combinations of j errors in the first group and $\begin{pmatrix} N/2-j \\ i-j \end{pmatrix}$ represents the total number of combinations of j errors in the second group.

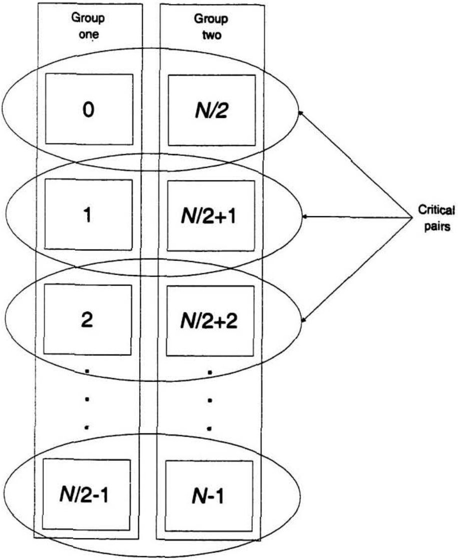


Figure D.3: Dividing SEs in the last stage into two groups.