VLSI DESIGN OF IMPROVED LOGICAL NEIGHBORHOOD NETWORK

by

ANJUM SHAIKH
B.E. (Electronics), N.E.D. University, Pakistan, 1984

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

We accept this thesis as conforming
to the required standard

[redacted]

Dr. F. Gebali, Supervisor, Dept. of Elect. & Comp. Eng.

[redacted]

Dr. A.K.S. Bhat, Member, Dept. of Elect. & Comp. Eng.

[redacted]

Dr. N. Djilali, Outside Member, Dept. of Mech. Eng.

[redacted]

Dr. S. Dost, External Examiner, Dept. of Mech. Eng.

Supervisor: Dr. F. Gebali

ABSTRACT

In the last few years the Internet and distributed computing has faced a spectacular growth. In the wake of the emerging telecommunications needs the Plain Old Telephone Systems (POTS) thus needed revolutionary steps to achieve economical and flexible networks. As a solution an optimum use of existing switching and transmission resources was achieved by integrating voice, data and multimedia applications within the same network. This solution was realized by applying packet switching techniques, which ultimately required data communication systems. The data communication systems necessitated high transmission Bandwidth and differentiated services (Quality of Service - QoS) networks. In the R&D industry the BISDN (Broadband Integrated service Digital Network) had emerged as the promising solution to these requirements, and is already replacing existing application-oriented communication networks.

The implementation of BISDN required development of network protocols, switching nodes, and transmission systems which could support the diversity of BISDN services. The IP and ATM became the most popular high speed network protocols. The advances in fiber optics technology have made available huge amounts of transmission bandwidth. Thus, in current networks the main cause of bottleneck is now due to the processing in the routers rather than the bandwidth of the transmission media. A switch design capable of supporting IP & ATM technology essentially requires a switching fabric within itself to effectively perform its functions. Any proposed switching fabric design must have high Quality of Service (QoS) capabilities and can support high data transfer rates to meet the performance requirements. One promising high speed switching fabric is the Improved Logical Neighborhood Network (ILN), which outperforms many other switching fabric architectures.

In this thesis, a design of ILN network and its hardware implementation is presented. The thesis first surveys different switching fabric architectures, with description of ILN network, and then compares the performance analysis of various networks. The proposed design for ILN network specifies hardware details and signaling conventions used. The hardware specifications elaborates the hierarchical modules of design,

comprising of a modular Switching Element (SE) node. The SE is an independent switching processor having all the desirable features such as distributed-routing (self-routing) using built-in routing algorithm, path uniqueness for each source-destination pair, and suitability for VLSI implementation.

In this work the hardware of ILN network is realized in the Field programmable Gate Array (FPGA) integrated circuit. The target technology applied is from Xilinx Virtex-II family. The functions of ILN network are demonstrated and verified by using simulation and synthesis Electronic Design Automation (EDA) tools. In the end of thesis the results of ILN network's hardware implementation are shown and discussed.

Examiners:

[redacted]

Dr. F. Gebali, Supervisor, Dept. of Elect. & Comp. Eng.

[redacted]

Dr. A.K.S. Bhat, Member, Dept. of Elect. & Comp. Eng.

[redacted]

Dr. N. Djilali, Outside Member, Dept. of Mech. Eng.

[redacted]

Dr. S. Dost, External Examiner, Dept. of Mech. Eng.

Table of Contents

List of Figures

List of Tables

Notation

ACK	Acknowledgement
AckIn	Acknowledgement In
AckOut	Acknowledgement Out
AddGen	Address Generator
ADMN	Augmented Data Manipulator Network
ATM	Aynchronous Transfer Mode
BISDN	Broadband Integrated service Digital Network
BRA	Bit-wise Routing Algorithm
CCITT	International Consultative Committee on Telegraphy and Telephony
CLK	Clock
CLP	Cell Loss Probability
DMN	Data Manipulator Network
DRA	Dynamic Routing Algorithm
DRC	Design Rule Check
EDA	Electronic Design Automation
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GCN	Generalized Cube Network
HDL	Hardware Description Language
HOL	Head Of Line
IADM	Inverse Augmented Data Manipulator network
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronic Engineers
ILN	Improved Logical Neighborhood Network
IM	(Switch) Input Module
IN	Interconnection Network

IP	Internet Protocol
ISE	Integrated Synthesis Environment
ISO	International Organization for Standardization
ITU	International Telecommunication Union
LNDMN	Logical Neighborhood DMN
LUT	Look Up Table
MAC	Medium Access Control
MIN	Multistage Interconnection Network
NACK	Negative Acknowledgement
NCD	Native Circuit Description
OAM	Operation And Maintenance
OM	(Switch) Output Module
OPSel	Output Port Selector
OSI	Open Systems Interconnection
PAR	Place and Route
PM2I	Plus-Minus 2^i connection (PM2I) network
POTS	Plain Old Telephone Systems
PrtGnt	Port Grant
QoS	Quality of Service
R&D	Research and Development
REQ	Request
ReqIn	Request In
ReqOut	Request Out
RTL	Register Transfer Level
SDF	Standard Delay Format
SDS	Space Division Switching
SE	Switching Element
SEI	Switching Element Input
SEO	Switching Element Output
SF	Switching Fabric
SM	Switching Module
SSM	Sub Switching Module

TDS	Time Division Switching
VHDL	Very High Speed Integrated Circuits (VHSIC) HDL
VLSI	Very Large Scale Integration

Acknowledgement

I am thankful to the almighty God, who is merciful and who gave me the strength and hope to accomplish.

First and foremost, I would like to thank my supervisor Dr. Fayez Gebali, for providing me the opportunity in achieving higher studies. His guidance put structure in my thoughts and gave me the will to continue. The atmosphere created by him in our group was well suited to innovation and the pursuit of excellence. Without all this support, my time at the University of Victoria would not have been a success.

I would also like to lend my appreciation to my fellow graduate students with whom I interacted during my time in Victoria. In particular I have extremely high regards for Mohamed Watheq El-Kharashi who extended his endless help and knowledge to me, whenever I needed the most. Here, I would also mention my gratitude for Abdullah Rehan and Sandeep Agarwal, with whom I used to had long discussions concerning to my studies.

Special thanks to Mr. Kirk Saban, Field Application Engineer, and Xilinx representative for providing his technical help and support.

My wife Samina and my son Adil were my vital source of encouragement and love. Samina, listened to my fears and worries; and, even when the going was tough, she never lost faith in me. Her encouragement and self-sacrifice provided me with the incentive and means to complete this project. For this, I am eternally grateful.

Dedication

To My Beloved Parents

Chapter 1

Introduction

1.1 Motivation

In this 21st century, the telecommunications industry is continuing to experience an extraordinary growth in its Data Communications field. By any measure, the growth is remarkable on all fronts: the number of hosts, the number of users, the amount of traffic, the number of links, the bandwidth of individual links, or the growth rates of service providers. This explosion of demand has occurred since last couple of decades due to rapid advances in computer and communication technologies. These systems requires to support distributed applications of voice, data, multimedia, control and supervisory. The technical interaction of these applications is achieved by implementing networks. The networks include switching systems and transmission media: point to point dedicated circuits, LANs, WANs, Virtual Private Networks, Intranets, Extranets, or the Internet [1, 2, 3].

An optimum solution to execute flexible and economical networks to meet the aforesaid demand is the integration of voice and data traffic over already existing switched communication networks. In the sequel, the Broadband Integrated Services Digital Network (BISDN) was accepted by CCITT (now ITU) as a standard [4] intended to be a worldwide Public Service Telecommunication Network (PSTN) to deliver a wide variety of integrated services. The developments in communication networks industry emerged with two most promising technologies, now being widely used: the Internet Protocol (IP) [5, 6] and Asynchronous Transfer Mode (ATM) [7, 8].

The implementation of networks supporting BISDN required huge transmission bandwidths and high speed switching processes within a communication system. The advances in fiber optics technology have made available the required amounts of trans-

mission bandwidths for BISDN. However, the design of a network that can support very high bandwidth switching services remains an ongoing challenge [9]. Thus, in current networks the main cause of bottleneck is due to the processing in the routers rather than the bandwidth of the transmission media. A switch design capable of supporting IP and ATM technology essentially requires a *switching fabric* component within itself to effectively perform its functions. Any proposed switching fabric design must have high Quality of Service (QoS) capabilities and can support high data transfer rates to meet the performance requirements [10].

There exists a large amount of literature that discusses various broadband switch architectures [11] - [20], but the majority talks about the theory and high level functions of proposed switches. However, there exists little detail on hardware complexities for developing a chipset capable of implementing a full *switching fabric* solution. This thesis targets a real world's digital logic design and hardware implementation of a *switching fabric.* In achieving the objectives of the thesis, the R&D work done in [21] - [24] contributes fundamental knowledge of BISDN switch designing; whereas, the theory and analysis works done for high performance switches in [10, 25, 26, 27] introduces a promising *switching fabric.* This later research work proposes the Improved Logical Neighborhood (ILN) network, which outperforms many other reported networks in the literature. The high performance of ILN network has motivated us to select this switching fabric for digital logic design and VLSI implementation.

1.2 Research Goals

The main research goals of this thesis are summarized as follows:

- study of communication networks, hardware designing, and strategies for synthesizable design,
- study of ILN switch fabric for a broadband switch,
- develop a digital logic model of the ILN switch fabric in VHDL environment,
- develop modular components of the model using top-down structured VLSI design methodology,

- implement hardware design of the model using Electronic Design Automation (EDA) tools.

1.3 Thesis Overview

Following this Introduction, Chapter 2 gives a global picture of a communication switch with its basic components. It discusses the various forms of queuing and buffering strategies in a switch. The chapter further focuses on the switch fabric component of switch, being the main topic of this thesis. In this thesis the switch fabric is referred as a communication network. The chapter thus discusses principles and constructions of various networks. The pros and cons of each network are discussed as well as their limitations.

Chapter 3 specifically discusses the theory of ILN network. The chapter then elaborates its basic building block - the switching element. This switching element is the hub of packet switching processes, and uses a built-in algorithm to route packet within the switch. The chapter discusses two routing algorithms known in literature as: Dynamic Routing Algorithm [26] and Bitwise Routing Algorithm [10]. Finally, the chapter reviews the reported performance of ILN network in the literature. This review also shows the performance comparison reports of ILN network with the other networks.

Chapter 4 provides specifications for ILN network design. The chapter initially discusses the design methodology and industry standards for designing a digital logic. The chapter then introduces the model for implementing ILN network theory. The model hierarchically shows the design from top layer to its basic component layers. Each layer gives detail of functions and required signals, according to the standard design methodology.

Chapter 5 presents VLSI implementation work of ILN network. It describes the applied implementation strategy by reporting the Hardware Design Flow. Further, the chapter defines the signaling conventions used in the design for control signaling between the interacting components. The chapter then exhibits the results of simulations, synthesis, and physical implementations of the design along with observations. In the end, the chapter discusses hardware optimization techniques being suggested

in the literature and were experienced during this work.

In Chapter 6 we conclude the thesis by providing a summary of the work, as well as some future directions for study.

Chapter 2

Communication Networks

2.1 Introduction

Communication networks are the fundamental means for interconnecting many different distributed applications. These systems provide media for information transfer between an end user to network, network to network and network to another end user. A communication network is a set of switches, interconnected by a set of transmission systems. The switching systems consists of nodes and subnets; where, the end users are connected with the subnets, and subnets with the nodes. The transmission systems provide physical connectivity of the network elements. These systems consists of transmission lines, line termination terminals, multiplexers, and repeaters. From Open Systems Interconnection (OSI) reference perspective the switching systems correspond to the data link and network layers; while transmission systems to the physical layer of the model [1, 2, 3].

As discussed in chapter 1, the BISDN is an objective standard in the telecommunications industry. The standard requires a high performance and an optimum communication network in the system that can meet the given benchmarks. The required performance of a network is determined by parameters such as capacity, bandwidth, and delay. In a network the measure of these parameters is mainly influenced by the behavior of involved switches and routers. The switching systems are the combination of various elements, which are responsible for the overall productivity of the system [10].

In this thesis we target hardware design of one of a component of the switching system. This chapter initially presents a global picture of a switching system, in section 2. The section describes switch structure and shows its constituent compo-

nents. The chapter in proceeding sections further reviews industry's popular design strategies for the target component family of the switch.

2.2 Switching Systems

A switching system performs a series of functions on incoming packets, which includes routing, buffering, control and management [5, 7]. In the industry large number of switch design alternatives exist, both as hardware and software modules. The software is essentially partitioned into hardware-dependent and independent components. However, in hardware the whole design is made modular for an efficient and scalable switch production [28, 29]. Since this thesis targets hardware design of a switch component; therefore, the further discussion is focussed on the hardware aspect of switch.

The components of the switch that have the most impact on its performance are the packet storage buffers or queues, and the switch fabric. Therefore, the switches can broadly be classified on the basis of two criteria: *buffering and queuing strategies*, and *type of switch fabric* . The following sub-sections outline a generic switch broadly showing switch components, switch's packet buffering strategies, and introduction to switch fabric.

2.2.1 Generic Switching Architecture

A generic switching architecture comprises of a set of modules, as shown in Figure 2.1 [10]. The input, output and switching fabric components provide the function of routing and buffering; whereas, the other components are responsible for control and management mechanism in the system. The switch is connected with its preceding and succeeding equipment through input and output module ports.

The **input module** (IM) performs the functions of: packet buffering, packet header error checking, storing and translation of valid header values, determination of the destination output port, passing relevant packets to control and management modules, addition of an internal tag containing internal routing and performance monitoring information for use only within the switch.

The **output module** (OM) prepares the packet streams for physical transmission

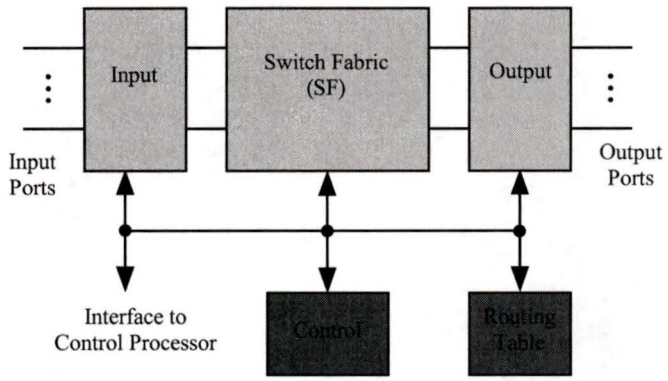


Figure 2.1. *Basic components of a switch*

by: processing and removing the internal tag, insertion of header values, possible mixing of cells from switch control and management modules with outgoing cell streams, cell rate decoupling, mapping packets to the lower layer protocol payload and generation of overhead.

The **control module** establishes, modifies and terminates connections and is responsible for: implementation of high-layer signaling protocols, interpret or generate signaling cells, negotiate traffic contracts with users, allocate switch resources for connections, including route selection, maintain and update routing table, admission or rejection decisions for requested new connections.

The control block further handles *management* functions of the switch system. It includes: lower or higher layer Operation And Maintenance (OAM), configuration management of switch components, security control for the switch database, usage measurements of the switch resources, traffic management, administration of a management information base, customer-network management, interface with operations systems and finally support of network management.

The **switch fabric** (SF) is the central functional block of the switch, which is responsible for routing the incoming cells/packets to the appropriate destination output port. The performance of SF is the major contribution towards the switch's overall characteristics. The cell switch fabric is primarily responsible for transferring of data cells and possibly signaling and management cells as well, between input and output modules of the switch. The other possible functions of switch fabric includes:

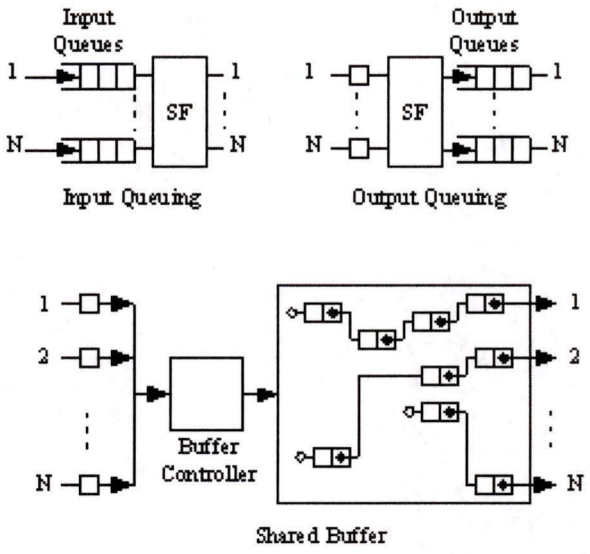


Figure 2.2. *Input, Output, and Shared Buffers*

cell buffering, traffic concentrating and multiplexing, redundancy for fault tolerance, multicasting or broadcasting, cell scheduling based on delay priorities, and congestion monitoring.

The main subject of this thesis is the hardware design of the *switch fabric* for a BISDN switch.

2.2.2 Buffering Strategies

Buffering is an important parameter in the switch design process. The location and size of buffers plays a vital role in the switch's performance, and is therefore planned in an optimized manner. Several switch designs are available in literature, exploiting different buffering strategies [30] - [35]. These designs employ three main approaches for providing buffering and queuing for high-performance switches: *Input Queuing/Buffering*, *Output Queuing/Buffering*, and *Shared Buffering*. These buffering strategies are illustrated in Figure 2.2. [10]

Input Queuing:

In input queuing systems a separate queue is dedicated with each input port of the switch fabric. This mechanism serves the purpose of buffering the packets on First-In-First-Out (FIFO) basis. This mean the incoming packets are stored at the tail

of the queue, and are served upon reaching the head of line (HOL). This system is feasible for VLSI implementation, since it does not need fast memory. However, it has the drawback of having broadcast and HOL problems; which eventually decreases the throughput of the switch. There are three potential causes of packet loss in this system: when input queue is full; during internal blocking within the switch fabric; and, when output queue is blocked.

Output Queuing:

In output queuing systems a separate queue is associated with each output port of the switch fabric. A small buffer is also required at the input of the switch to receive and process the incoming packets. The controller reads incoming packet's header and routes the packets through switch fabric to appropriate output queues. This system suffers from HOL problem, just like input queuing. The solution is to increase the operating speed of switch fabric and the output queue, which could lead to complexity in hardware implementation, e.g. using Double Data Rate (DDR) memories. There are four possible causes of packet loss in this system: when input buffer is full; during internal blocking within the switch fabric; when output queue is blocked; and, when output queue is full.

Shared Buffer:

A shared buffer design employs a single common buffer, and does not require a switch fabric for routing. The packets are initially buffered at the input, and then the switch controller queues them within the shared buffer by applying a linked list data structure. A linked list contains the addresses of packet storage memory locations. These locations further contains the address of data for next successive packet storage memory location. Ultimately, last such memory location has the pointer directing towards a specific output port. If N is the number of input and output ports, the shared buffer performs $2N$ times read and write operations per slot, and thus operates at $2N$ line speed. There are two possible causes for packet loss in shared buffer system: when input buffer is full; and, when shared buffer is full.

The design for the Switch Fabric (SF) being hardware implemented in this thesis, is expected to be used in a switch design which would implement both input and output queuing strategies.

2.2.3 Switch Fabric

In telecommunication service industry the switching fabric is also known as *Interconnection Networks (IN)*. The INs were initially investigated in multiprocessors and distributed shared-memory multiprocessors . These systems were first introduced in telecommunications in digital switches for wire-line telephones, but with the advent of packet switching for data communication they became dedicated for use in digital networks [36] - [40].

The switching fabrics (or INs) routes user packets from one of its input port to the desired output port by using either *Time Division Switching (TDS)* or *Space Division Switching (SDS)* techniques. Thus the INs are broadly divided into these two types of systems.

In TDS systems a switching fabric functions with a speed equal to the sum of the input rates and serves incoming packets sequentially at its output. This is achieved by using a shared communication medium for the flow of incoming cells across input to output ports. This communication medium is a shared bus or a shared memory, being accessed by users through Medium Access Control (MAC) protocol. The scalability of time-division switches is limited by the bandwidth of the shared medium and the centralized controller requirements. Shared medium architectures do not scale well and can only support a relatively small number of ports. Space division switching supports high performance switches as they provide high capacity at low access time delay.

In SDS systems a switching fabric applies an interconnection scheme that provides many routes across the fabric, allowing several packets to be directed to distinct output ports concurrently. This system uses interconnected mini-switches for establishing straight or cross path, to form a network. The incoming cells flow through this network from an input to an output port. These networks are broadly subdivided into *crossbar networks* and *multistage interconnection networks* (MIN). There was a little development in the crossbar systems, after it was realized its hardware implementation is cumbersome and they make it difficult to provide guaranteed qualities of service [41, 42]. However, in the field of multistage interconnection networks there is a continuous process of progress since last 30 years.

In this thesis, the target design network (switch fabric) belongs to the class of

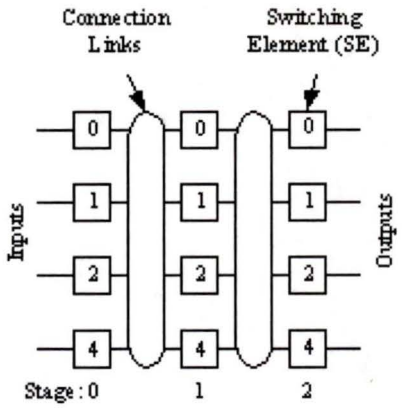


Figure 2.3. *A 4 × 4 MIN with 3 stages and 4 switches per stage*

MIN, therefore the following sections focuses their discussion over the networks of this family.

2.3 Multistage Stage Interconnection Networks

A Multistage Interconnection Network (MIN) , is a subclass of indirect networks, being created by the combination of nodes [43] - [52]. Each node in the network is a crossbar switching element (SE). The SEs in MIN are usually identical and have been traditionally organized as set of stages. These SE stages are knitted with each other by connection links to create a routing path in a certain topology. The inputs or outputs of this network are termed as ports. A 4 × 4 MIN with 3 stages and 4 SEs per stage is shown in Figure 2.3 [10]. The number of ports are denoted by *'N'*, the number of switches by *'w'*, and the number of stages by *'n'* being calculated as $n = \log_2 N$. These networks have different properties depending on the number of ports, size of each SE, number of stages, and the interstage connection pattern.

In MINs, the number of stages and the SE connection patterns between stages determine the routing capability of the MINs. Different SE connection patterns give different characteristics and topological properties of MINs, leading towards varied MIN classes and functions.

2.3.1 MIN classes

The MINs are classified in terms of two different approaches: First, depending on the type of channels and switches; and Second, depending on the availability of paths to establish new connections. MINs have been traditionally sub-divided into three classes: Blocking, Nonblocking, and Rearrangeable MINs [36, 37].

Blocking: A connection between a free input and output pair may not always be possible, because of conflicts with the existing connections. Typically, there exists a unique path between every input and output pair, thus minimizing the number of switches and stages in hardware. Such networks are economical, but are vulnerable to internal blocking of packets.

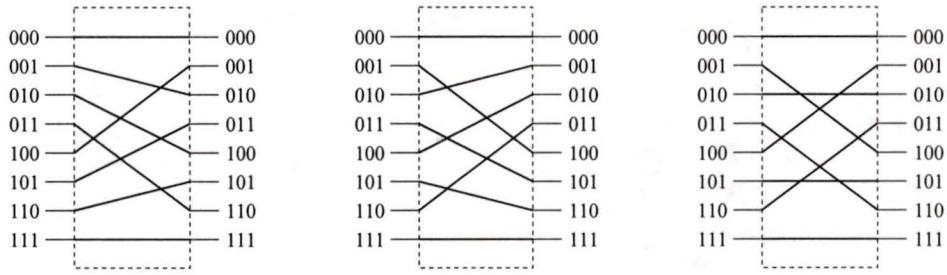
Nonblocking: Any input port can be connected to any free output port without affecting the existing connections. Non blocking networks have the same functionality as a crossbar. They provide multiple paths between every input and output, thereby offering high throughput. However, they are costly hardware components due to the requirement of extra SE stages.

Rearrangeable: Any input port can be connected to any free output port. However, the existing connections may require rearrangement of paths. These networks also require multiple paths between every input and output but the number of paths and the cost is smaller than in the case of nonblocking networks.

2.3.2 MIN Functions

A $N \times N$ MIN has a one-to-one correspondence between its inputs and outputs, and the connections thus established are also called *permutations*. A Permutation Network is a MIN capable of performing all $N!$ one-to-one mappings of its N inputs to its N outputs; thus generating several networks. There are six commonly known permutations: Shuffle connection, Inverse Shuffle connection, Exchange connection, Butterfly connection, Cube connection, Plus-Minus 2^i connection [36, 37].

Shuffle Function: The perfect shuffle connection function S performs a cyclic left shifting of the binary bits of a number A to the left for one position:



(a) Shuffle Function (b) Inverse Shuffle Function (c) Exchange Function

Figure 2.4. *The shuffle, inverse, and exchange connection for $N = 8$.*

$$\begin{aligned} A &= a_{n-1}\ a_{n-2}\ a_{n-3}.....a_1\ a_0 \\ S(A) &= a_{n-2}\ a_{n-3}.....a_0\ a_{n-1} \end{aligned}$$

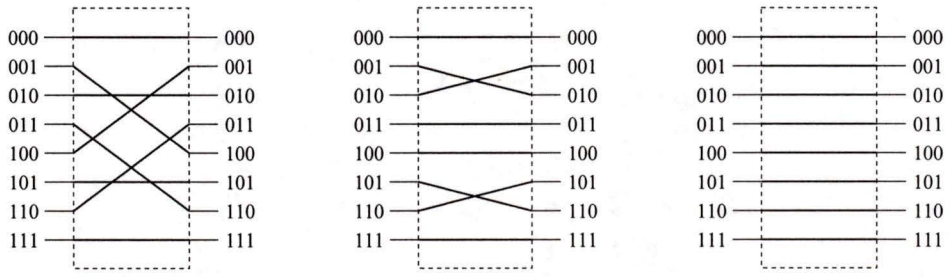
Example: for $S = 2$ (010), the function will left-shift the bit pattern one position to the left, thus creating $S(A) = 100$ (4), as demonstrated in Figure 2.4 (a) [37].

Inverse Shuffle Function: The inverse shuffle does the opposite to shuffle function, i.e., the function S^{-1} performs a circular right shift on the binary bits of number A:

$$\begin{aligned} A &= a_{n-1}\ a_{n-2}\ a_{n-3}.....a_1\ a_0 \\ S^{-1}(A) &= a_0\ a_{n-1}.....a_2\ a_1 \end{aligned}$$

An example is shown in Figure 2.4 (b) [37].

Exchange Function: This permutation is usually referred to as bit reversal; the function $E^{i,j}$ exchanges or interchanges the bits at positions i and j leaving all other bits intact:



(a) 2nd Butterfly (b) 1st Butterfly (c) 0th Butterfly

Figure 2.5. *The butterfly connection for $N = 8$.*

$$\begin{aligned} A &= a_{n-1}...a_i...a_j...a_0 \\ E^{i,j}(A) &= a_{n-1}...a_j...a_i...a_0 \end{aligned}$$

An example is shown in Figure 2.4 (c) [37].

Butterfly Function: The butterfly function B^i interchanges the least-significant bit and the ith bit of a binary number A leaving all other bits intact:

$$\begin{aligned} A &= a_{n-1}...a_{i+1}\ a_i\ a_{i-1}...a_0 \\ B^i(A) &= a_{n-1}...a_{i+1}\ a_0\ a_{i-1}...a_i \end{aligned}$$

The straight one-to-one connection is also called identity connection I. An example is shown in Figure 2.5 [37].

Cube Function: The cube function C^i complements the ith bit (a_i) of a binary number A leaving all other bits intact:

$$\begin{aligned} A &= a_{n-1}...a_{i+1}\ a_i\ a_{i-1}...a_0 \\ C^i(A) &= a_{n-1}...a_{i+1}\ \bar{a}_i\ a_{i-1}...a_0 \end{aligned}$$

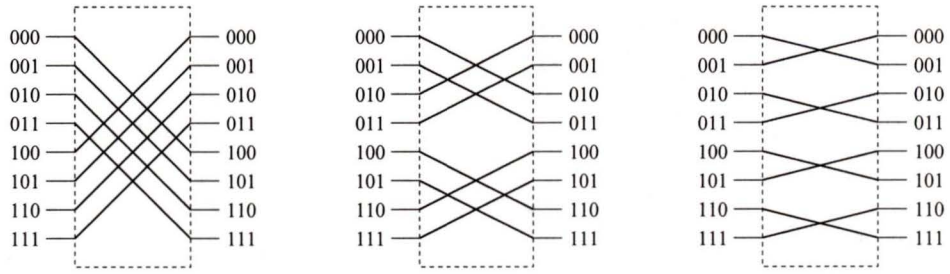


Figure 2.6. *The cube connection for $N = 8$.*

Figure 2.6 [37] shows the cube connection for i = 0,1, and 2 with $N = 8$. C^0 is also called the exchange connection.

Plus-Minus 2^i Function: In the Plus-Minus 2^i connection (PM2I) Network the switching elements are arranged in stages within *Plus-Minus 2^i (PM2I)* distance and are connected according to the interconnection function defined by:

$$PM2_{+i}(P) = P + 2^i \ modulo \ N \quad for \ 0 \leq i < m$$
$$PM2_{-i}(P) = P - 2^i \ modulo \ N \quad for \ 0 \leq i < m$$

where,
N = number of input/output ports = 2^m
m = number of switching element stages = $Log_2 N$
i = switch stage address
PM = plus-minus
P = address of switching element (SE), $0 \leq P < N$

These functions provide link connection address of the next stages SE for the source SE. This is being done by adding or subtracting 2^i from the source SE addresses. This means, it allows SE P to send or receive data from any one of SE $P + 2^i$ or SE P - 2^i, arithmetic modulo N, $0 \leq i < m$. Figure 2.7 [36] shows the $PM2_{+i}$ interconnections for N = 8. The function $PM2_{+i}$ adds 1 to the ith bit position of the addresses and the function $PM2_{-i}$ subtracts 1 from the ith bit position of the ad-

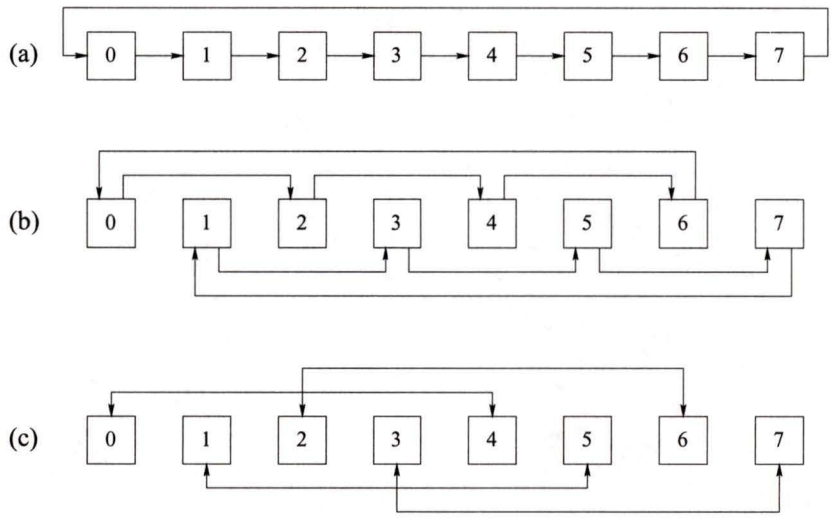


Figure 2.7. *PM2I network for N = 8. (a)$PM2_{+0}$, (b)$PM2_{+1}$, (c)$PM2_{+2}$*

dresses (arithmetic modulo N). Diagrammatically, $PM2_{-i}$ is same as $PM2_{+i}$ except the direction is reversed.

MINs have desirable features and enjoyed intensive research and development. Various new networks based on MINs are being developed, studied and reported in literature including in [38] - [52]. In [38] all such innovative networks are placed into a perspective family tree; and is reproduced here in Figure 2.8. The design of ILN switch fabric for this thesis is based on the MIN hypothesis; thus all such related successive network designs are discussed in the following sections.

2.4 Generalized Cube Networks (GCN)

The *Generalized Cube Network (GCN)* is a multistage cube-type network topology and was introduced by Siegel and Smith in 1978 [10, 36, 46]. The network has N inputs, N outputs, $m = \log_2 N$ stages, where each stage consists of a set of N links connected to $N/2$ four-state *switching elements*. The number of links between stages is N. The connections in this network are based on the shuffle and cube functions, as discussed in sub-section 2.3.2. The name generalized-cube network refers to the network consisting of the generalized-cube topology and four states or functions of switching element (SE).

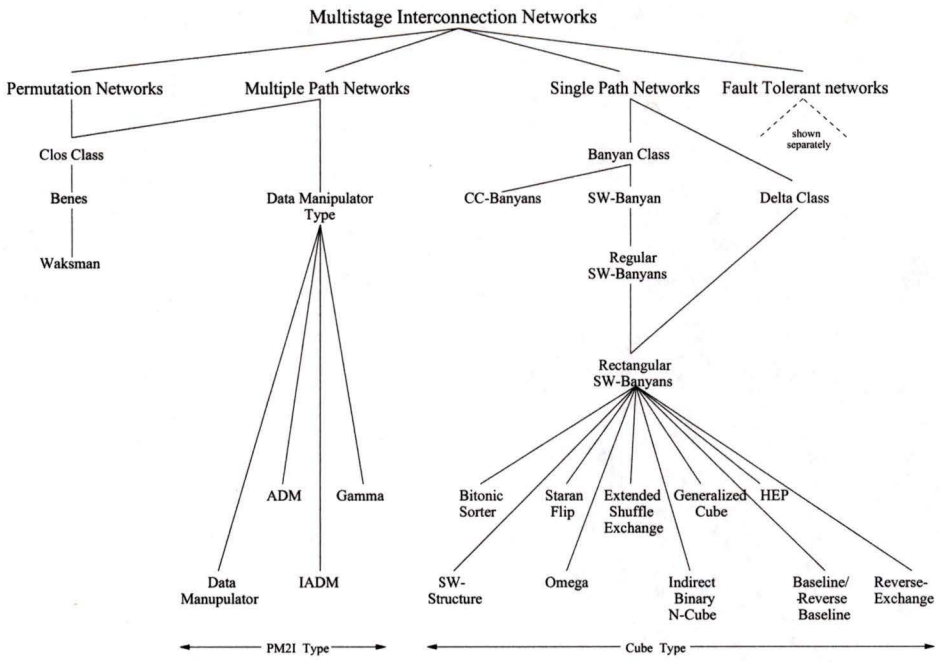


Figure 2.8. *Multistage Interconnection Network family tree*

2.4.1 The GCN Topology

The GCN topology is shown in Figure 2.9 [10] for $N = 8$. In the figure, the labels of the links entering the upper and lower inputs of SEs are used as the labels for upper and lower outputs, respectively. The labels are the integers from 0 to N - 1.

The generalized cube network is a blocking network and provides only one path from any input to any output. As such, it possesses no tolerance for faults. Refer GCN figure, the switching element SE(i,j) at stage i and position j is connected to SE(i+1,k), such that k is given by [10]:

$$k = \begin{cases} j & \text{straight connection} \\ C^i(j) & \text{cube connection} \end{cases}$$

2.4.2 GCN's SE Functions

The *switching element* (SE) is a two input and two output crossbar switch, and is set to one of three legitimate states. These states are:

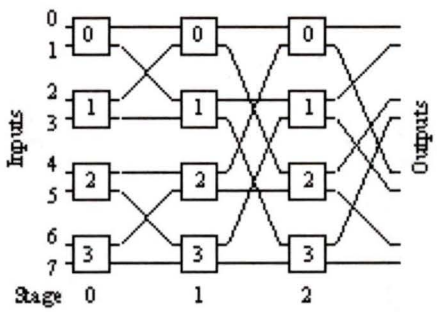


Figure 2.9. *Generalized Cube Network topology for $N = 8$.*

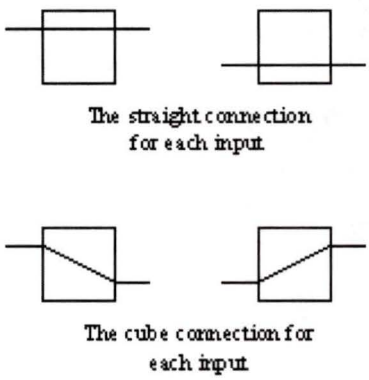


Figure 2.10. *The straight and cube connections of an input SE in GCN*

1. Straight; i.e., input 1 to output 1, and input2 to output 2

2. Upper cube connection; i.e., input 1 to output 2

3. Lower cube connection; i.e., input 2 to output 1

These three states are set by the control structure of a network according to the cube interconnection function, and are shown in Figure 2.10 [10].

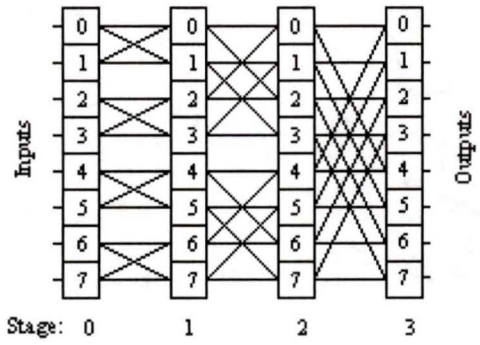


Figure 2.11. *An 8 × 8 Banyan network*

2.5 The Banyan Networks

Banyan networks, so called because its wiring is said to resemble the roots of a Banyan tree, are a class of single-path blocking MINs. That is, in Banyan structure there is only one path from any given input to any given output. As such, it possesses no tolerance for faults. This network was first proposed by Goke and Lipovski in [53] which defined the Banyan network in terms of its graphical representation.

Banyan networks are very suitable for the multi-processor interconnection due to their cost-effectiveness. These networks serve the dual purpose of partitioning the resources in the system and providing communication between various nodes in the system. The self-routing property of Banyan networks provides great simplicity in the control of the switching elements and hence makes them attractive for implementing high speed switching nodes. Many blocking networks have been developed which are topologically equivalent to the Banyan networks, out of which two are significant. These are: Siegel and Smith's Generalized Cube Network and Wu and Feng's Baseline Network [38, 47, 48].

A Banyan Network definition can be interpreted as shown in Figure 2.11 [10]. The network comprises of N-input × N-output ports having interconnected 2-input × 2-output identical switching elements (SE). These SEs are arranged in $n+1$ stages, where $n = \log_2 N$. The number of SEs in each stage is N, and the number of connecting links between stages is $2N$. The SEs present in network's input stage are used as 1-to-2 multiplexers; whereas, the ones in network's output stage are used as 2-to-1 concentrators. The SEs present in other internal stages of the switch fabric utilizes

both of their input and output ports. In the network the switching element SE(i,j) at stage i and position j, is connected to SE(i+1, k) such that k is given by [10]:

$$k = \begin{cases} j & \text{straight connection} \\ C^i(j) & \text{cube connection} \end{cases}$$

where $0 \leq i < n$; providing one unique path from any input to any output based on the input row address and the destination address.

The Banyan networks belong to the class of blocking networks because of the non-zero probability of finding a path between two nodes. Once an intermediate node is in use for some other connection the network gets blocked, for particular ports. Due to this reason, the Banyan network cannot be directly used for switching purposes. In order to redress this bottleneck various topologies based on the Banyan network have been studied and proposed in the literature [9]. The work on these proposed architectures has thus mainly focused on the performance enhancement of the Banyan fabric with no significant work on the resource utilization or scalability of the architecture.

One of the known Banyan network is a *Batcher-Banyan* [1]. This network make use of the property that a Banyan network becomes internally non-blocking if the inputs are sorted with respect to the destination requests. This is accomplished by putting a switch in front of the Banyan switch to permute the cells into a configuration that the Banyan switch can handle without loss. The sorting switch used is a Batcher switch, invented by K.E. Batcher in 1968. The first Batcher-Banyan ATM switch was designed by Huang and Knauer in 1984 and is known as Starlite. The other known Batcher Banyan are Moonshine by Hui in 1987; and Sunshine by Giacopelli et al. in 1991. A *Rectangular-Banyan* [38] is one in which the number of inputs into each node and the number of outputs from each node are equal. An another particular Banyan network is the *SW-Banyan* network [9, 38, 39, 47]. It is basically a rectangular Banyan, created recursively by expanding a crossbar structure, with two inputs and outputs each at every node.

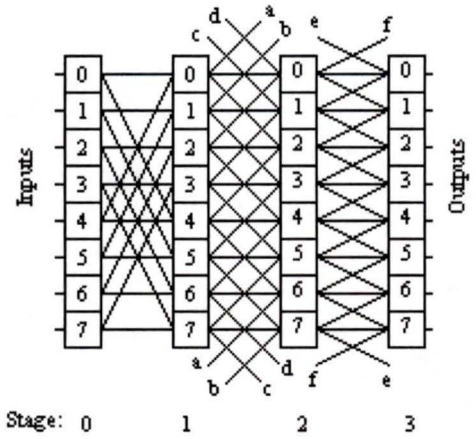


Figure 2.12. *The DMN and ADMN for N = 8*

2.6 The Data Manipulator Networks

The class of MIN called Data Manipulator Networks are based on the PM2I interconnection functions [36, 54]. The DMNs are a wired series of PM2I interconnection functions. These networks were further enhanced to Augmented Data Manipulator (ADMN) [39], the Inverse Augmented Data Manipulator(IADM) [25], and the Gamma Networks [38, 50]. Since the target design of this thesis is based on DMN and ADMN networks, therefore only these two networks are discussed in the following sub-sections. An 8×8 DMN/ADMN network is shown in Figure 2.12 [10].

2.6.1 Data Manipulator Network (DMN)

The Data Manipulator Network (DMN) was introduced in [54]. A typical model of DMN has N input and output ports, N SEs arranged in a stage and addressed from 0 to N-1, $n = \log_2 N + 1$ stages of switching elements (SE). The switching stages are ordered from n - 1 to 0. The SEs has 3 input and output ports - the SEs present in input stage are used as 1-to-3 selectors and in output stage are used 3-to-1 selectors. The SE's internal interconnection functions at stage i are $PM2_{+i}$, $PM2_{-i}$, and straight. Each SE selects one of its input links and connects either to one of its output links, or to two or three of its output links; i.e., the nodes can connect either in one-to-one or in a broadcast setting.

As shown in figure 2.12, an actual network implementation may have only two distinct outputs from each node in stage m-1. In this network the switching element SE(i,j) is connected to SE(i+1,k) such that k is given by [10]:

$$k = \begin{cases} (j - 2^{n-i-1}) & \text{-2I (up) connection} \\ j & \text{s (straight) connection} \\ (j + 2^{n-i-1}) & \text{+2I (down) connection} \end{cases}$$

The controls of the DMNs are limited to one pair per stage. At stage i, SEs whose ith address bit is '0' responds to one set of controls; SEs whose ith bit is '1' responds to the other set of controls. This means, stage i of the network can receive any two of the three signals [36]:

$$H_0^i \text{ or } H_1^i,$$
$$U_0^i \text{ or } U_1^i, \text{ and}$$
$$D_0^i \text{ or } D_1^i.$$

where,
H = horizontal, corresponds to straight connection,
U = up, corresponds to -2^i modulo N connection,
D = down, corresponds to $+2^i$ modulo N connection.

SEs whose ith address bit is '0' are controlled by the signals with a '0' subscript, and those whose ith address bit is '1' are controlled by the signals with a '1' subscript.

2.6.2 Augmented Data Manipulator Network (ADMN)

The Augmented Data Manipulator (ADM) network is a modified DMN, wherein the word *Augmented* means: 'greater than before' [36, 39]. The ADM network was introduced in [44] with two enhanced features in DMN:

- the use of independent control for SEs; i.e., each SE can be independently set to straight across, -2^i mod N, or $+2^i$ mod N, and

- the use of a dynamic routing tag scheme; i.e., each SE in the network can derive its own control signals from information included in the routing tag.

These features provided ADM networks an edge over DMN of having an improved fault-tolerance. The ADM networks has the ability to avoid faulty and busy SEs or links, by exploiting its multiple-path property and does dynamic re-routing of connections.

The ADMN, as shown in figure 2.12 [10], is a blocking network that provides two alternative paths from any input to any output. Thus, this network is 1-fault tolerant. The ADMN has good potential for VLSI implementation due to its low density, regular wiring, and modularity. The switching elements (SE) used in this network are 3-input $\times$ 3-output crossbar switches. In the network, the input stage SEs acts as 1-to-3 multiplexers, the output stage SEs as 3-to-1 concentrators, and all other internal stages SEs as regular functioning crossbar switches.

If the stages of ADM network are traversed in reverse order, the resulting network is called *Inverse Augmented Data Manipulator (IADM)* [36]. In this network the interconnection functions of the input stage are $PM2_{+/-0}$, the interconnection functions of the next stage are $PM2_{+/-1}$,...., and the interconnection function of the output stage are $PM2_{+/-(m-1)}$.

2.7 Summary

In this chapter we discussed communications networks, which are the essential elements in the Data Communications field. It is further observed that switching systems are the backbone of a communication network. In this scenario the switching system and their essential components required to build a switch remained the focus of study in the rest of the chapter.

The study of switching systems encompassed a global view of switch's functional blocks and their working strategies. The main block of switch: a generalized Switch Fabric is discussed exclusively, being the objective of the design for this thesis. The switch fabric's classes and functions are thus reviewed in length. The study zoomed into the DMN and ADM Networks, being the desired networks to cope industry

requirement.

The study of literature reveals that while further enhancing the features of the DMN family the Research and Development work has introduced an another interesting network known as *Improved Logical Neighborhood Network (ILN)*. The ILN network is found to be a very productive switch component in terms of QoS parameters. It is thus being the topic of this thesis, and is therefore discussed exclusively in the next chapter.

Chapter 3

The Improved Logical Neighborhood Network (ILN)

3.1 Introduction

The Improved Logical Neighborhood (ILN) network is an outcome of industry's continued impetus in research and development for a high QoS, high fault tolerance and efficient switching systems. In the literature several broadband switches are being proposed using different switching fabrics and buffering strategies. These design decisions have a major impact on the switch performance and hardware complexity. The switching fabrics proposed include buses, Banyan and Crossbar based networks [9]. The research also proposes the design of switches using switching fabrics in combination with input and output buffering strategies. This is being suggested to take advantage of the desirable properties of both queuing approaches for achieving high QoS parameters [33, 67, 68]. Among the various switching fabrics reported in the literature, the ILN network has been observed to be an efficient and fault tolerant than other known networks [25, 26, 27]. In this thesis the ILN network is thus chosen for study and VLSI implementation.

The ILN switching fabric belongs to a class of multistage interconnection networks (MIN) and is based on the concepts of Data Manipulator Networks (DMN) [25]. In research it was revealed that DMN has the potential of providing the desired networks, due to its inherent available multi-paths from an input to the output ports. Further studies resulted the advent of Augmented Data Manipulator Networks (ADMN) [36]. A continued research provided more improvements to ADMN and proposed a Logical Neighborhood DMN (LNDMN) solution. The newer version to the LNDMN is then

introduced in research studies as the Improved Logical Neighborhood network (ILN) [25]. This switching fabric is required to be utilized in a broadband switch which would be using input and output queuing buffers in its design architecture.

This chapter discusses the ILN Network's theory, it's constituent components, and the review of performance study being already done in research and development work. Note that the chapters of this thesis use the terminology *network* for the switching fabric of switch. The initial section of this chapter instigate the ILN network concept and its working principles. The main part of chapter then focuses on the central component of ILN network: the *switching element.* In the review of switching element, the chapter specifically discusses the function of packet routing process in the SE. At the end of this chapter, the performance of ILN network is discussed and reviewed.

3.2 The ILN Network Theory

The Improved Logical Neighborhood Network (ILN) was initially introduced in [25].The function of ILN switch fabric is to route the arriving packets at its input port to its desired destination output port, by employing Space Division Switching (SDS) techniques. The ILN network system is capable of providing desirable fault tolerance and QoS features. This new network is a N-input $\times$ N-output ports switching fabric. The network is comprised of cascaded and independent functioning crossbar switching processors called *Switching Elements* (SE). These SEs are arranged in the network in columns and interconnected with each other at a Hamming distance of '0' or '1'. This arrangement of SEs in columns of the network is termed as switching *stages.* An eight input-output ILN network is shown in Figure 3.1 [10].

The calculation of SE stages in ILN network is done by considering N as the number of equal input and output network's ports. With this consideration, the number of stages in ILN network are n+1 where $n = \log_2 N$. Each stage has a column of N Switching Elements (SE). The SEs present in the input stage are used as 1-to-4 multiplexers; in the output stage as 4-to-1 concentrators; and, in the other internal stages as regular and identical 4 $\times$ 4 crossbar switches.

The ILN network does no packet buffering and provides $n!$ alternating paths be-

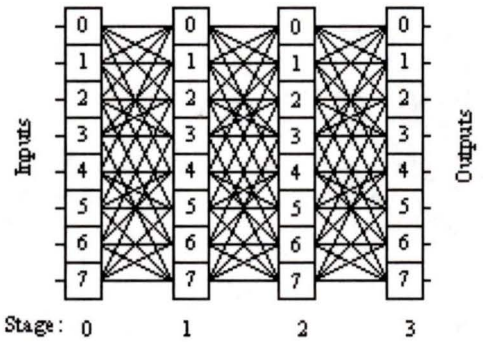


Figure 3.1. *An Improved Logical Neighborhood Network for 8 ports*

tween any input and any output ports. The network is very fault tolerant and the blocking probability is much smaller than other MIN networks. The interconnection scheme of SEs in ILN network is done on the basis of cube connection function [36], being already reviewed in the previous chapter. Each SE has a specific address in terms of its location in the ILN network. In Figure 3.1 consider i as the stage index and j as the vertical location within a stage of SEs; thus the address of a particular SE in ILN network will be SE(i,j). A Switching Element SE(i,j) is connected to a next stage SE(i+1,k) within ILN network in a manner such that k is given by [10]:

$$k = \begin{cases} j & \text{straight connection} \\ C^0(j) & \text{0-cube connection} \\ C^1(j) & \text{1-cube connection} \\ C^2(j) & \text{2-cube connection} \end{cases}$$

where $0 \leq i < n$. The Switching Element (SE) is the main building block of ILN network, which performs the switching processes independently. Since in ILN network, all switching processes are done at the SE, therefore its working is described in detail in the following sections.

3.3 The Switching Element

The function of Switching Element (SE) is to update its own address location within the ILN network; and then, while idle be ready to receive an arriving packet's request,

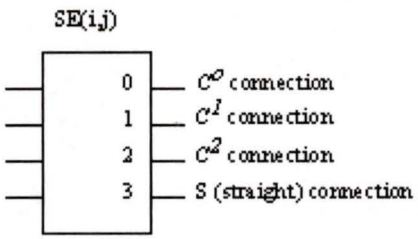


Figure 3.2. *The Switching Element icon*

identify it's destination address, and route the packet data after establishing a distinct path. The SE is fault tolerant and is also responsible to resolve any contention for its output ports, due to incoming packet requests at its input ports.

The SE is a 4 input and 4 output port crossbar switch. An icon of SE is shown in Figure 3.2 [10]. The figure shows SE's ports and output connections with the next stage SE. The output connections of SE with the next SE in ILN network are done according to the cube connection function. The $C^{0,1,2}$ identifies 0,1,2-cube connections, and S as straight-connection, respectively. The SE performs its function with the help of its constituent functional modules and the built-in routing algorithm. The routing of packets within the SE is carried out in two phases: first, *path establishment* and then, *cell transfer*.

In the **path establishment phase**, a routing table or vector is created using the available routing algorithm within the SE. The link selection criterion is then based on this created routing table. The function works on the principle of request (REQ) and acknowledgment (ACK) protocol. Each input port having a packet request, to be routed, selects an available output port, according to the available links in routing table, and sends a REQ to the next SE. In case of congestion on selected path the system seeks alternate paths until all routing options are tested. In case no alternate path is available the system responds to its preceding SE, with a negative acknowledgment (NACK). However, a positive acknowledgment is always propagated backwards till it reaches the originating port indicating a path establishment.

In the **cell transfer phase**, actual packet data transfer takes place, along with an assertion of FLAG signal, indicating a valid data present on data wires, to the next SE.

3.4 The SE Routing Algorithms

The switching element (SE) establishes path for the arriving packet to the requested destination port of ILN network with the help of its built-in routing algorithms. The function of routing algorithm is to create a routing table or vector in accordance to the arriving packet's request at an input port of the SE. This routing table is used by the SE for connecting its certain input port with a specific output port. Since each SE's individual output port is connected with a particular next stage SE, the input-output port connection of SE thus establishes a unique path leading towards the desired destination ILN network output port. There are two known routing algorithms: the *Dynamic Routing Algorithm* (DRA) published in [26], and, *Improved Bit-Wise XOR Routing Algorithm* (BRA) devised in [10]. Both of these algorithms are reviewed next.

3.4.1 The Dynamic Routing Algorithm (DRA)

The Dynamic Routing Algorithm (DRA) [26] establishes a connection path between an originating SE with the next stage SE in a distributed manner, to connect with the destination port of ILN network. The DRA on receiving the destination address from the preceding equipment generates a routing table 'P' to select a potential next SE, connected with one of the 4 output ports of present SE.

The routing table P is generated on the basis of two concepts: one, the cube connection or Hamming distance concept of the ILN network, and second, the concept of early routing. The *Hamming distance* concept outlines that two SEs in adjacent stages are connected if their Hamming distance is minimum; i.e., the chosen output port will connect the minimum origin and destination SE addresses. The *early routing* criteria outlines that two SEs in adjacent stages are connected if the path has the minimum vertical distance to the destination. This means if a path produced by Hamming Distance property, has identical routing options then the path which provides early routing will be selected by the system.

In general, the Hamming distance $\mathcal{H}$ between SE *S(i,j)* and destination y is defined as $\mathcal{H}(S(i,j), y)$. Similarly, the least vertical distance $\mathcal{V}$ between address k of a SE in stage i, and destination y is defined as $|k - y|$. According to the DRA principle, the

algorithm will create the routing table P by applying Hamming Distance $\mathcal{H}$ criteria first, and then Vertical Distance $\mathcal{V}$ concept so that $P(\mathcal{H}, \mathcal{V})$ is minimal.

Consider a packet routing example to understand DRA while taking into account three definitions. First, the Hamming distance property defines that the distance between two adjacent SEs is the binary address difference among them. Second, the vertical distance means the absolute difference between the integer addresses of SEs in two adjacent stages. Third, the ILN network principle defines that each SE's output is connected with the next SE on cube connection basis, i.e. in ILN network the SEs are interconnected at a binary distance of '0' or '1'. Refer figure 3.2, a SE having j address = '000' will have $Cube^0$ output connected to '001', $Cube^1$ output connected to '010', $Cube^2$ output connected to '100', and Straight output connected to '000' next SE j addresses.

Consider, SE (j = '000') receives a request for destination address y = '111'. Now, the $\mathcal{H}/\mathcal{V}$ for y at the output identified by $Cube^0$ connection = 2/6, at $Cube^1$ connection = 2/5, at $Cube^2$ connection = 2/3, and at Straight connection = 3/7. The DRA will thus arrange these values in ascending order as 2/3, 2/5, 2/6, and 3/7 to create the routing table P. This arrangement is done while accounting minimum $\mathcal{H}$ first, and then the minimum $\mathcal{V}$ distance. Now according to the P the SE will select first choice route through $Cube^2$, second choice route through $Cube^1$, third choice route through $Cube^0$, and fourth choice route through Straight connection output ports. This same process of generating P and route selection criterion will repeat at SEs of each stage until the desired output port of ILN network is accessed.

3.4.2 Bitwise Routing Algorithm (BRA)

The Bitwise Routing Algorithm (BRA) [10] also establishes path between originating and destination SEs in a distributed manner. However, it does not involves arithmetic operations; instead it relies upon bit-wise XOR operation on the destination SE's address D and the originating SE's row address j. This results in generating a routing vector p, which carries the routing information of destination path, and is given by [10]:

$$p = XOR(j, D)$$

The routing vector p is updated at each stage within the succeeding SE, until all its bits are corrected to zeros indicating the path is established to the destination row address. The theory of this algorithm is that the vector p carries information about the mismatch between the originating address and the destination address. Each stages SE is able to correct a one bit, being assigned '1', located at any position in p. Thus, the mismatch would decrease by one bit each time the arrival packet's path establishment request passes through a stage of the network.

Consider an example that exhibits a maximum routing scheme for the possible SE outputs, used to establish the path based on the numerical value of routing vector p. Assume the originating SE address is "010" and the arriving packet's destination address request is also for "010" SE, then the created routing vector will be $p = 000$. The SE routing system will thus only choose its straight connection. On the other hand consider if the originating SE address is "000" and the arriving packet's destination address is "111" port, then the created routing vector will be $p = 111$. The SE routing system will now test and access its all cube connection choices in the ascending order.

The algorithm also employs an *intelligent routing technique* to counter any anticipated blocking of path due to port contentions or poor usage of resources. These solutions are taken into consideration while foreseeing the scenarios of: First, two packet requests attempting for a same destination port; Second, there is only one straight connection option available but the desired path has faulty or busy equipment (links or next SE); Third, when in the initial stages there exists low alternate routing priorities, but there is sufficient redundancy available in the next stages of network. The intelligent routing technique suggests to redress the mentioned problems by corrupting the already existing few '0' bits of the routing vector p to '1'. This technique will thus prevent an unnecessary congestion at a particular stage, allowing the system to choose alternate paths to reach the desired output. That is to say, the actual vector p generation process is delayed by one stage. The intelligent routing technique then transmits both correct and corrupted address to the successive SE,

with the confidence that the corrupted bits of p will be returned back to zero value at the later stages.

In this thesis, the design of ILN network incorporates the Bitwise Routing Algorithm in its SE components. This routing algorithm was chosen because of its simplicity and ease of implementation. However, the algorithm's intelligent routing technique is not fully implemented to avoid any complexities in the design.

3.5 ILN Network Performance Review

The high performance networks are weighted by measuring their Quality of Service (QoS) parameters. Typical QoS parameters are: average packet delay, cell loss probability, throughput, and fault tolerance. In the literature these parameters are analyzed with different approaches. A review of results showing ILN network performance measurement is studied in this section.

The terminal reliability of ILN network has been calculated in [25], for a N input and output ports network, and is given by the equation:

$$R_t = (1 - q^{m+1})^{m+1}$$

where,
R_t = Terminal Reliability,
$m = \log_2 N$,and
q = Failure Probability.
In accordance to this relation, the performance of ILN network is measured for N = 8, 16, 32 and 64, and is shown in Figure 3.3. The figure shows that the reliability decreases slowly as the probability of failure increases. Further, for a given failure probability, the reliability increases as the network size increases. This is advantageous, since larger switches have higher manufacturing cost and should be more immune to total failure. Furthermore, the ILN network has reliability of almost a 1 for failure probability of 50% when $N = 64$.

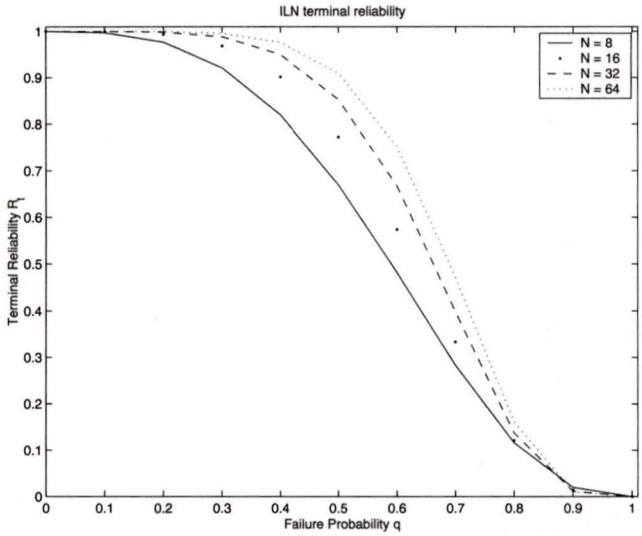


Figure 3.3. *Terminal Reliability of ILN network when* N = *8,16,32,64*

The performance of ILN network is studied in [26] by conducting numerical simulations to analyze the cell loss probability and throughput. The behavior of network is observed by varying the number of input or output ports (N), by varying output FIFO buffer sizes, and by keeping input FIFO buffer sizes fixed. The packet arrival traffic load is assumed to be high at 90%, with no network internal faults. The result in terms of cell loss probability (CLP) and throughput parameters are shown in Figures 3.4 and 3.5. In the figures, it is depicted that cell loss probability of 10^{-6} can be reached when Output-Buffer = 45; and throughput of 100% when Output-Buffer = 35 for all network sizes.

The performance of ILN network is analytically studied in [10] in terms of throughput and average delay. The relations devised in this study are reproduced and discussed below.

An $N \times N$ ILN network is being built by using (n + 1) × (n + 1) crossbar SEs, in the n - 1 internal stages, 1-to-(n + 1) selectors in the input stage and (n + 1)-to-1 concentrators in the output stage; where n is the number of SE stages in ILN network. The probability p(i) that i packet requests arrive at a time slot addressed for an output port is given by:

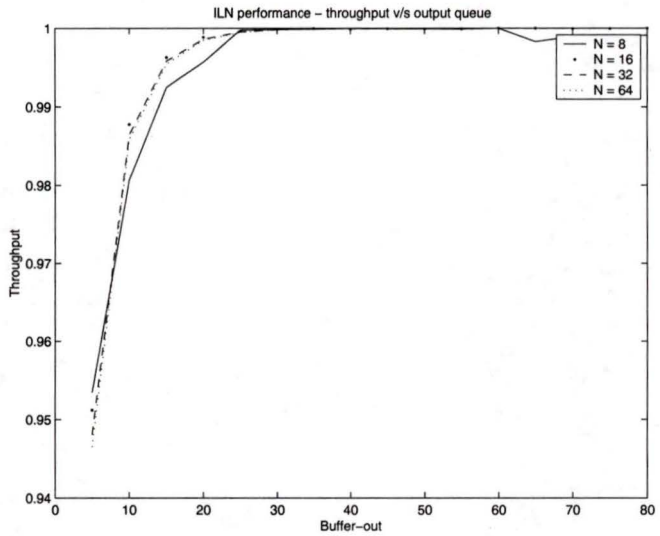


Figure 3.4. *The ILN network throughput performance results - from [26]*

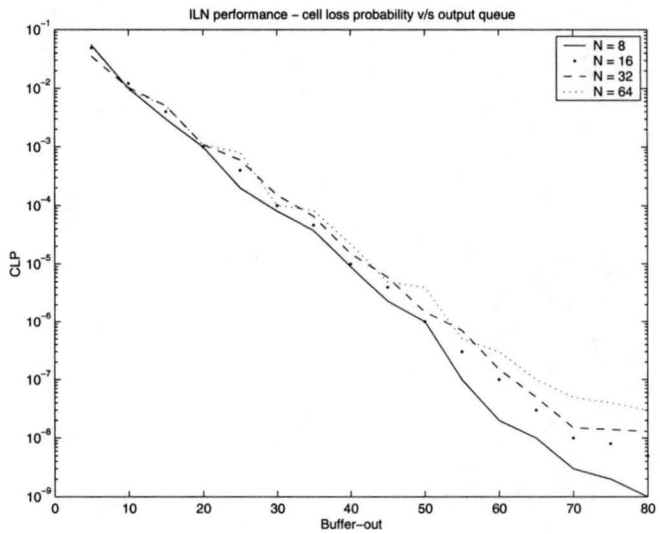


Figure 3.5. *The ILN network CLP performance results - from [26]*

$$p(i) = \binom{N}{i} \left(\frac{a}{N}\right)^i \left(1 - \frac{a}{N}\right)^{N-i}$$

where, a is the arrival probability at an input port. The average number (or flow) of packets that arrive at the inputs per time slot is given by:

$$N_{avg} = E[ip(i)] = \sum_{i=0}^{N} ip(i)$$

The throughput is defined as the average number of cells which are successfully transferred between the input and output ports of the switching fabric per time slot. The formula of Kruskal and Snir [45] for the throughput Th is

$$Th = \frac{q}{1 + qs(b-1)/(2b)}$$

where, q = a/(n+1) is the arrival probability at any internal link, s = n - 1 is the number of internal stages, and b = n + 1 is the number of links in the internal SEs. Thus, the throughput is given by:

$$Th = \frac{a}{n + 1 + an(n-1)/[2(n+1)]}$$

The above formula is true when the SEs in the last stage can only accept packets for one of its n + 1 inputs. If the SEs in the last stage can accept all n + 1 packets in one time slot, then the throughput would be

$$Th = \frac{a}{1 + an(n-1)/[2(n+1)^2]}$$

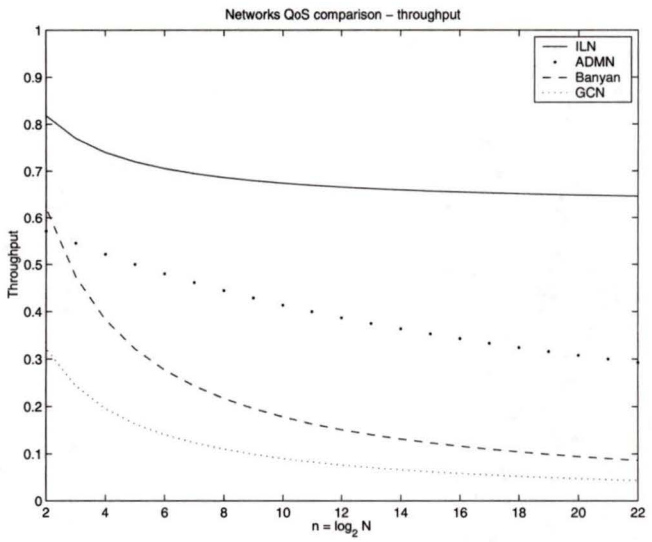


Figure 3.6. *ILN network throughput comparison with other Networks - from [10]*

The probability that a packet will access the desired output, p_a, is defined as

$$p_a = \frac{Th}{N_{avg}}$$

The delay of the network is defined as the average time a packet spends in the network before it succeeds in accessing an output port. The average delay time, t can be calculated with the expression

$$t = \frac{1 - p_a}{p_a}$$

The above equations for throughput and average delay are applied to compare the performance of ILN, ADM, Banyan, and the Generalized Cube networks using MatLab simulations. These results are obtained by applying 90% traffic load, and are shown in Figures 3.6 and 3.7. The figures depict that the performance of other networks in comparison to ILN network declines for throughput as the network size increases. However, the ILN network performance in terms of Delay deteriorate after

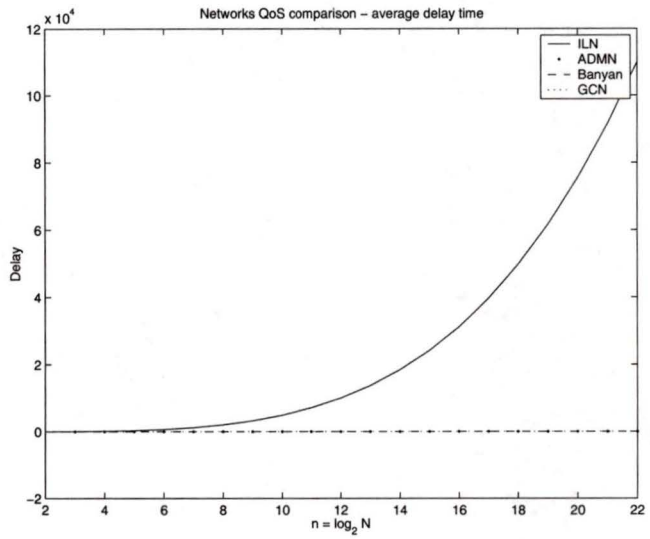


Figure 3.7. *ILN network delay comparison with other Networks - from [10]*

n = 6 (N = 64). This is due to the time consumption for path establishment and then cell transfer phases during the packet switching process. This could be a trade-off between throughput within network or a delay at higher size of networks. However, in practical switches the size of typical networks is within the range of $N = 64$. Since, the ILN network performance over given QoS parameters are significantly high than most networks, thus it is the most desirable switching fabric.

The performance of the ILN network is also being analyzed in the [27], by applying numerical simulations. The results of this study has verified the relations devised in [10]. The study exhibits that the ILN network's performance, and the devised routing algorithms are correct. The study has also proved with statistics that the ILN network has outperformed the other networks over throughput, average delay, and cell loss probability QoS parameters.

3.6 Summary

In this chapter we have specifically overviewed the Improved Logical Neighborhood (ILN) Network. We have defined the ILN network theory from [25] and explained the applications in terms of its functional modules. The ILN network is comprised of

interconnected switching nodes called Switching Elements (SE). The SE is the main building block of ILN network and it's working is localized once activated. The SE has self routing capabilities for packet transport from origin to destination ports, while using its built-in packet routing algorithms. The network uses distributed routing at its each node and establishes path between SEs using its redundant links.

The chapter further reviewed the ILN network theory's relevant routing algorithms and explained them with examples. In the end of the chapter a review of ILN network reliability and performance analysis, being previously done in the literature, was presented. This review included the comparison of ILN network performances with other known networks.

It has been observed in this study that ILN network is a non-blocking switch fabric since it allows $n+1$ packets to traverse from its any input to any of its output ports. The network is fault tolerant as it performs satisfactorily in presence of n faulty links and nodes. The results of analysis has proved that the ILN network out-performs other networks in terms of high standard QoS parameters: high throughput, low delay, and low CLP. The network has thus high capabilities for handling multicast, broadcast and hotspot traffic for a switch.

Since the ILN network has all desirable qualities required by ATM and IP switches for data communications, therefore it has been chosen in this thesis as the model for hardware implementation. The next chapter draws digital design specifications of ILN network, based on the theory discussed in this chapter.

Chapter 4

Digital Design of ILN Network

4.1 Introduction

This chapter presents a digital design model of ILN network, based on the theory discussed in the previous chapter. The functions of ILN network described in the previous chapter are taken into account in terms of hardware implementation. Thus this model provides necessary specifications for the VLSI implementation of the ILN network, being presented in the next chapter. The chapter outlines the design methodology and then details the ILN network's technical functional building blocks, along with the strategies for routing of packets within the switching fabric.

The chapter describes the ILN network in a top-down fashion, and devises digital specifications for the hardware implementation of the network. Section 4.2 describes the design methodology used for modeling the ILN network. This section section reviews the VLSI industry standards and approach for integrated circuits (IC) designing process. Section 4.3 outlines the top layer of design, which is the ILN network. The section specifies a complete ILN network scheme, explaining the connection links tieing the constituent switching processors of the network. Section 4.4 details the main building block of ILN network - the switching element (SE). This section identifies all the composite layers of SE and their interconnection with each other. The sections 4.5, 4.7, and 4.8 provides specifications for each layer of SE, with details of all applied control and data signals. The section 4.7 discusses an important block of SE, where actual packet switching mechanism is housed. This section further details the module's control and functional components interaction in the sub-section 4.7.3. The chapter ends with a summary of technical specifications for the ILN network.

4.2 Design Methodology

The success of a design of any integrated circuit depends on the use of a structured design strategy. This strategy uses systematic methods to simplify the design process. The advantages of structured design are the provision of options for easier management of the design complexity, which eventually allows flexibility in making design changes and simplification in the verification and debugging processes. A number of structured design techniques are known in VLSI industry but all have great deal of commonality. The structured design strategy follows four steps: hierarchy, modularity, regularity, and information hiding [55, 56, 57].

The proposed design of the ILN switching fabric of this thesis follows the structured design strategy.

4.2.1 Hierarchy

Hierarchy uses the idea of dividing a functional module into layers of simpler sub-modules, and then repeating this operation on the sub-modules until the complexity of the basic module represent an appropriately comprehensible simple functional unit. This strategy is accomplished by using a set of abstractions developed in the industry to describe integrated electronic systems. In this scheme the design is divided into three distinct domains: behavioral, structural, and physical domains.

The behavioral domain specifies what specific functions are being performed by a system; e.g., by a top layer icon representation. The structural domain specifies the subdivided entities, and the manner they are integrated together through interfaces, being well defined and fully specified; e.g., by block diagram. Finally, the physical domain specifies how to actually build a structure that has the required connectivity to implement the prescribed behavior of the model. In the IC design, a well-defined behavioral, structural, and physical interface indicates the layer, name, layer type, size, and signal type of external interconnections, along with logic function and electrical characteristics.

4.2.2 Modularity

The principle of modularity is to create modules in a design, which serve several functions. These modules are thus to be designed once and then they are used several times within the same design, or in different designs. The advantage of this strategy lies in the reduced design time, the need to verify the module only once, and allowing for easier design changes. In hardware design, during simulation and synthesis [detailed in next chapter], in order to accomplish physical design level the technology design library files are utilized. These design files contain commonly used components, being the basic building blocks of a design, such as registers, counters, multiplexers, ALUs, etc.

4.2.3 Regularity

The concept of regularity is to create consistency in the design. The designs fulfillment of this requirement eases the placement and wire routing tasks of VLSI fabrication process. This is achieved by considering the regular module shape and layout, routing, and regular signal transition. The regular module shape means dividing the design into set of a similar building blocks. The shape and layout of the modules refers to the size of the modules and the location of physical interconnections of their different signals. For example, I/O ports could be placed on the left and right of the module with control signals applied from the top and bottom of design icon. The regular signal transition also points to using of a single edge of a clock instead of using both edges in a same process.

4.2.4 Information Hiding

Information hiding suggests creation of black box modules, whom internal construction is hidden from rest of the circuit. This means, only the behavioral description of the module is considered in the design. The use of such modules simplifies reviews, modifications and debugging of the design. One example could be of using a memory element in the design, with no detail of its internal construction, except its external signals and the response of the module.

Table 4.1. *The ILN network control and port signals*

Signal	Type	Description
Clk	In	Switch system's Clock input signal - the entire ILN system is synchronized on this global signal.
Reset	In	Asynchronous reset signal.
NIn	InOut	ILN input packet transport ports: 0 to 7; Each port contains in & out control signals, and input packet data [31:0] lines.
NOut	InOut	ILN output packet transport ports: 0 to 7; Each port contains in & out control signals, and output packet data [31:0] lines.

4.3 The ILN Network

The function of ILN network is to initialize and perform switching by routing the incoming packets from any of its input port to the desired output port. The network is required to provide highest QoS and Fault Tolerance features. The ILN network model described in this design is an $N \times N$ switch fabric, where N is fixed at 8, for simplicity. As per theory of ILN network, the model uses 4×4 identical and modular crossbar switches called switching elements (SE), which are inter-connected to each other to make a network, in a cube connection fashion. The SEs are arranged in $n + 1$ stages, where $n = \log_2 N$. The stages of the network individually consists of a column of N SEs. In the network configuration each switch has an *(i,j)* address, where i indicates horizontal and j as vertical location of SE.

An ILN network scheme, with connection links and connected SEs, is shown in Figure 4.1, and a description of signals is shown in Table 4.1. The Figure 4.1 is a detailed diagram for the ILN network, that was schematically shown in Figure 3.1 of previous chapter. The blocks in Figure 4.1 are all identical and represents a switching element (SE). Each SE at (i,j) ILN network address is labeled in binary form as shown. The constituent SEs of ILN network has input-output ports of packet transport: data and signaling, along with the control signals: Reset-In (RI), Reset-Out (RO), SE Address-In (AI), and SE Address-Out (AO), respectively. Within ILN network, the

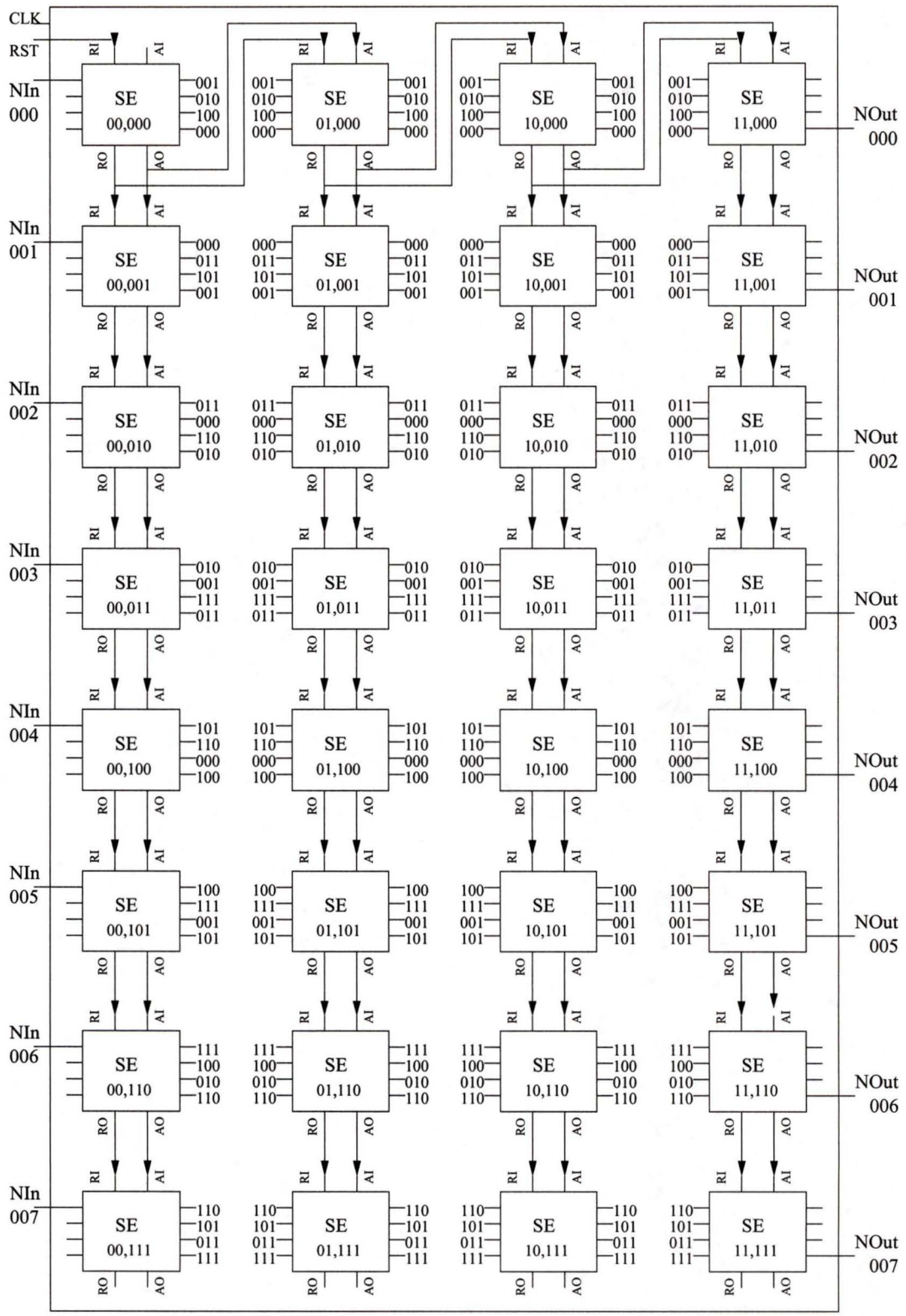


Figure 4.1. *The ILN Network Structure*

SEs are connected with each other through links, according to the cube connection function. Note that the input and output packet transport links of a specific SE of the ILN network are showing the addresses of preceding and succeeding equipments connected to the SE.

Refer Figure 2.1 of chapter 2 showing a generic switch system having input (IM) and output (OM) modules along with the switch fabric. The switch system is responsible to provide necessary Reset and Clock signals to the ILN switch fabric. In Figure 4.1 the Reset input of ILN network is connected to its constituent 0th SE; whereas, the Clock input is a global or synchronous signal for all components of network. The network initializes its system upon assertion of Reset signal. During initialization, the SEs of ILN network generates their own address, and provides control reset signal and address information to their successive SEs within the switch fabric. At the ports of ILN network, the input (IM) and output (OM) modules of the switch implements FIFO queues that are able to buffer incoming and outgoing packets to and from the other layers [refer OSI model], during the switching functionality of ILN network. As described in Table 4.1 each port of the ILN network is a pipe of wires that contains control and data signals. The IM and OM performs a sequence of interactive signaling (both-way) in a hand-shaking protocol over control wires of NIn and NOut ports of ILN network, during the path establishment phase. However, during the actual packet transfer phase the flow of packet is in one direction from inport to the outport over data wires of NIn and NOut ports of the network.

The ILN network performs its functions with the help of its constituent modular elements, according to the structural design strategy. A hierarchy of the ILN network's top-to-down design is shown in Figure 4.2. The specs and functions of these individual modules are discussed in the following sections.

4.4 The Switching Element

The switching element (SE) is a modular and the main building block of ILN network. The SE consists of 4 input and 4 output ports, and implements a built-in distributed routing algorithm. The routing algorithm used in this design is Bit-wise Routing Algorithm (BRA), discussed in Section 3.4.2. The design does not employ

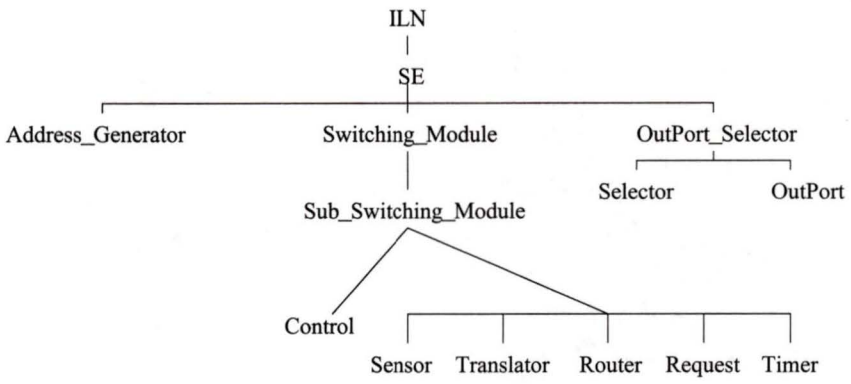


Figure 4.2. *The ILN Network Hierarchy*

the hardware complexity for the implementation of *intelligent routing* aspect of BRA. This approach in the design is taken into consideration to have a simple and better understanding of the ILN network theory and hardware implementation processes. However the design exhibits all the properties of ILN network including SE's contention resolution aspect.

The design of SE is required to fulfill all the following functions of a fundamental SE:

1. at the initialization of system, update SE location within the ILN network,

2. provide SE's own addressing information to the neighboring SEs in the ILN network,

3. provide initialization signal to the SE's neighboring SEs within ILN network,

4. initialize SE's internal functional modules,

5. when ready after initialization, process packet switching and implement contention resolution according to the ILN network principles.

The SE should be able to continue its functions even with its faulty ports or connected

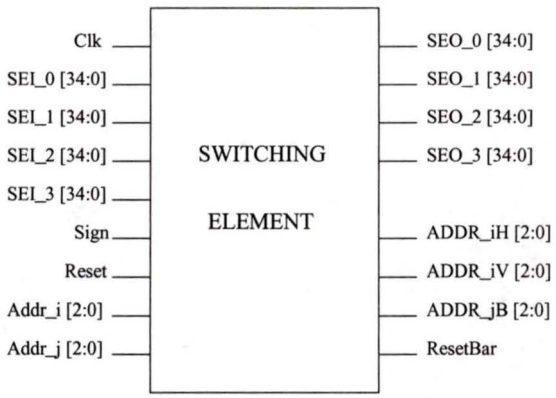


Figure 4.3. *A top level SE diagram*

links at its ports; until all its resources are exhausted. This gives graceful degradation of the network in the presence of faults and provides the system with a high degree of fault tolerance.

A top layer of SE with its input and output ports is shown in Figure 4.3. The details of SE signals are described in Table 4.2. In the figure, the clock (Clk) is a global input signal to the ILN network system; all events of ILN network are synchronous with its rising edge. The Reset is an asynchronous signal input to ILN network, and upon its assertion (active-low) the network's initialization and switching process activates. The switch inputs (SEI-0, SEI-1,..) and switch outputs (SEO-0, SEO-2,..) signals are the pipes or bundles of wires, meant for packet control and data signal transportation. The other auxiliary input and output signals provide addressing and control information being required during the initialization process of SE.

The SE performs its functions by employing smaller and modular functional hardware components. The composition of SE constitutes three modules: an *Address Generator (AddGen)*, a *Switching Module (SM)*, and an *Output Port Selector (OPSel)*. The internal structure of SE is shown in Figure 4.4. In this figure it is also shown that SE provides some internal signals for its constituent modules. The internal signal *Enable* triggers the switching function of SE at ready state, after performing the initial addressing function. This signal is meant to be asserted after a certain delay, during which all other ILN network SEs perform the addressing function to update their location addresses and reach the ready state. The other internal signals are address j

Table 4.2. *Top Level SE signal description*

Signal	Type	Description
Clk	In	Clock input: all SE events are synchronized at the rising edge of this clock signal.
Reset	In	Asynchronous reset input signal to trigger SE functions.
Sign	In	SE Control Input signal - when '1': SE assigns its address as 0th, when '0': SE calculates its address with respect to its preceding SE's Address
Addr-i [2:0]	In	Preceding SE's Stage Address - its 0th bit signifies the address is coming either from Vertical or Horizontal ILN location source: when = '1': preceding SE at vertical, when = '0': preceding SE at horizontal.
Addr-j [2:0]	In	Preceding SE's ILN vertical location address.
SEI-0 [34:0]	InOut	SE Packet Input at port number 0.
SEI-1 [34:0]	InOut	SE Packet Input at port number 1.
SEI-2 [34:0]	InOut	SE Packet Input at port number 2.
SEI-3 [34:0]	InOut	SE Packet Input at port number 3.
Addr-iH	Out	SE's calculated address of its stage, for the neighboring Horizontal SE - 'H' signifies Horizontal: 0th bit = 0.
Addr-iV	Out	SE's calculated address of its stage, for the neighboring vertical SE - 'V' signifies Vertical: 0th bit = 1.
Addr-jB	Out	SE's calculated address of its vertical ILN location - 'B' signifies Both, being same for both vertical or horizontal neighboring SE
ResetBar	Out	Asynchronous reset output signal to neighboring SE's functions. It is asserted when valid output address signals are available.
SEO-0 [34:0]	InOut	SE Packet Output at port number 0.
SEO-1 [34:0]	InOut	SE Packet Output at port number 1.
SEO-2 [34:0]	InOut	SE Packet Output at port number 2.
SEO-3 [34:0]	InOut	SE Packet Output at port number 3.

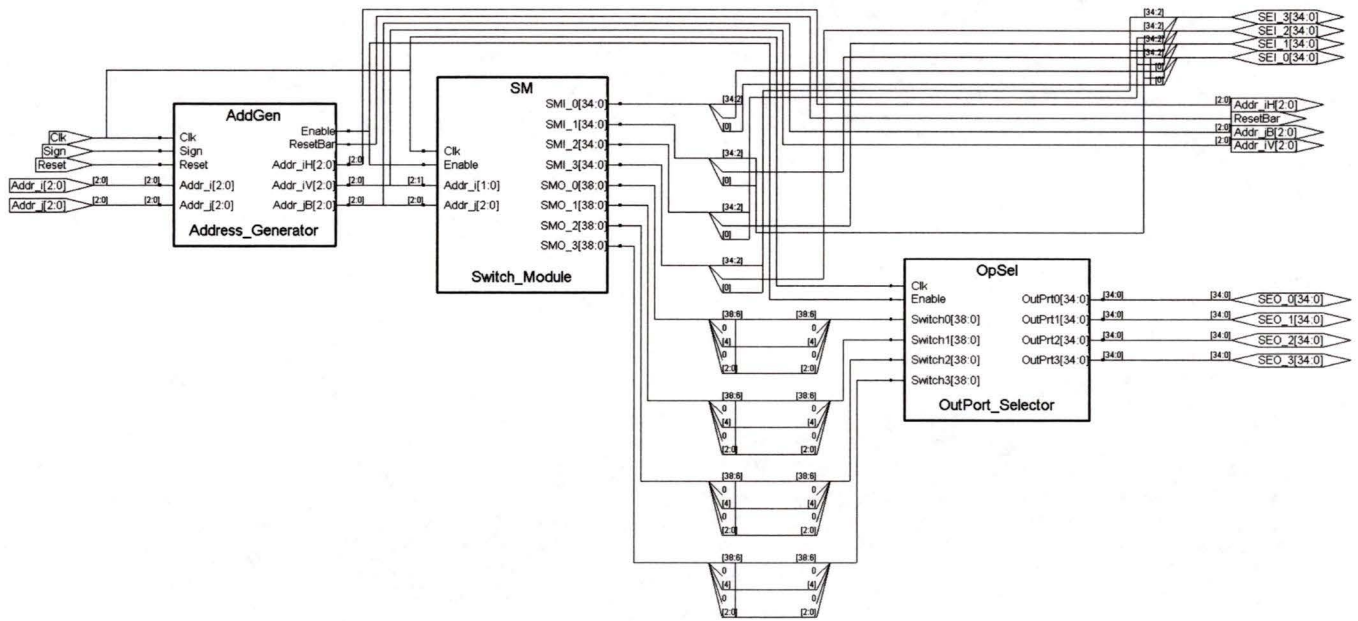


Figure 4.4. *The structure of Switching Element*

branched from outgoing address *Addr-jB* signal, and address i signal extracted from the 2 MSB bits of outgoing address *Addr-iV* signal. These i and j signals signifies the SE(i,j) address location within ILN network.

In the ILN network the Reset input is connected with the SE at location (0,0) i.e., the top left switch of the network. The Sign input signal of SE(0,0) is tied to '1' (declaring it to be the 0th SE in ILN network), whereas the other SEs Sign input is kept '0' (declaring them to be the non 0th SE in ILN network). The SEs of the ILN network, performs addressing function at initialization, upon assertion of Reset signal, while considering the value of the input Sign signal. This addressing information is then transmitted to the SE's internal SM module and to the neighboring horizontal or vertical SE. In the ILN network, the top row of the SEs provide addressing information to the successive horizontal neighboring SE's Addr-i and Addr-j inputs, through horizontal addressing signals (Addr-iH & Addr-jB). Each top SE in-turn provide addressing information to the successive vertical neighboring SEs downwards in each ILN network stage, using vertical addressing signals (Addr-iV & Addr-jB). In the same fashion the ResetBar signal of preceding SE is connected to the Reset signal of succeeding SE. This ResetBar signal is asserted, once a valid address is produced over Addr-iH, Addr-iV, and Addr-jB output signals.

The detail specification of SE functions performed by its constituent elements are described in the following sections.

4.5 The SE Address Generator

The function of Address Generator (AddGen) is to create SE's location address in the ILN network, provide this address info to the SE's other internal components, and to the neighboring horizontal and vertical SEs. Further, the address generator provides Enable signal to the SE's internal functional modules after a certain delay. This delay is equivalent to the required number of clock cycles needed by other SEs of the network to perform the addressing function.

The Address Generator performs its functions by utilizing the input initialization and addressing signals of the SE. The resulting values of these functions are then produced at the output ports of the Address Generator module. An icon of the

Address Generator is shown in Figure 4.5, and its signals description are shown in Table 4.3.

A flow diagram showing sequence of Address Generator functions is shown in Figure 4.6; and the detail of these functions are discussed below.

The Address Generator module triggers upon assertion of Reset signal, which signifies start of SE's initialization and availability of valid 'address' and 'sign' signals at its input. The module generates SE address after examining the value of Sign signal. If the Sign signal is pulled to a '1', the Address Generator creates 0th i and j addresses. If Sign signal is at '0', it extracts the previous SE's i value from the two MSB bits of input Addr-i signal and determines Addr-i 0th bit. In case this bit is equal to '0', it is perceived by the system that this SE is located at horizontal location with respect to the preceding SE. Thus extracted previous SE's i address is incremented by one to create the SE's i address; whereas, the j address is copied as is from the input Addr-j signal, to create the SE's j address. However, if the input Addr-i signals 0th bit equals to '1', it is perceived by the system that this SE is located at vertical location with respect to the preceding SE. Thus received Addr-j value is incremented by one to create j; whereas, the extracted i remains unchanged to create the SE's address, respectively.

The Address Generator module also creates output address i-Vertical (Addr-iV), address i-Horizontal (Addr-iH), and address j-Both (Addr-jB) signals for neighboring SEs in ILN network. The Addr-iV is a 3-bit vector, created by concatenating a '1' at 0th bit place and generated i value at the MSB of the vector. Similarly, a 3-bit Addr-iH is created by concatenating a '0' at 0th bit place and the generated i value at the MSB of the vector. In ILN network one of these signals are connected with the Addr-i input of the neighboring horizontal or vertical SE. The 3-bit addr-jB vector contains the created j signal value. In ILN network this signal can be connected to the Addr-j input of both or any horizontal and vertical neighboring SEs. After the generation of i and j output signals of SE, the Address Generator asserts ResetBar output signal, signifying valid address signals available at the output of SE. This ResetBar signal is attributed by the next SE as an input Reset signal.

The module's next function is to provide an output Enable signal, which serves as an input Enable signal for the other internal components (SM and OPSel) of the SE.

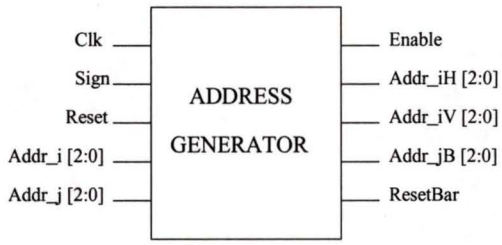


Figure 4.5. *The SE Address Generator*

Table 4.3. *Address Generator signal description*

Signal	Type	Description
Clk	In	Clock signal
Sign	In	Indication signal - when '1', the SE is at 0th location
Reset	In	When asserted signifies start of SE initialization, and valid address at Addr-i, and Addr-j inputs.
Addr-i [2:0]	In	0th bit identifies position of SE with respect to the preceding SE, and MSBs provides i address
Addr-j [2:0]	In	SE j address
Enable	Out	Enable signal for other internal components of the SE, and is asserted after a certain delay.
ResetBar	Out	Reset signal for the neighboring SEs
Addr-iH [2:0]	Out	Address i for the neighboring horizontal SE.
Addr-iV [2:0]	Out	Address i for the neighboring vertical SE.
Addr-jB [2:0]	Out	Address j for both vertical and horizontal neighboring SEs.

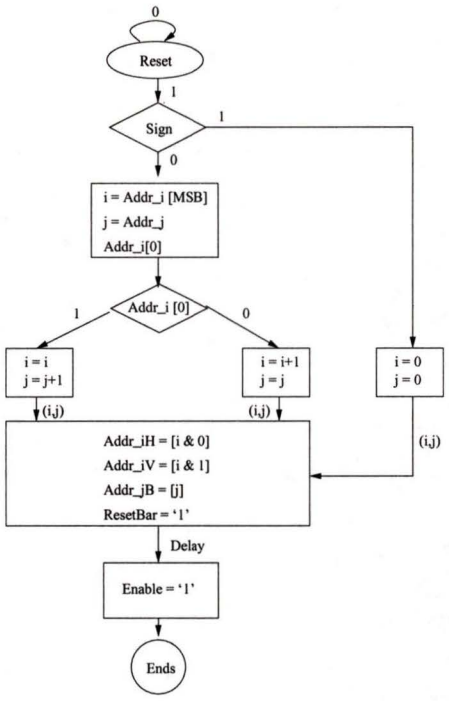


Figure 4.6. *Address Generator functional flow diagram*

The Enable signal is asserted after a certain *Delay*, equal to the number of clock cycles required by other ILN network SEs to perform their addressing function. During the simulation phase of ILN network's hardware implementation (discussed in next chapter), it has been observed that each SE's Address Generator module requires a maximum of 3 clock cycles period to perform its functions. With this identified result, the required *Delay* can be calculated as:

$$Delay = 3 \times (Network\ Size\ -\ SE\ Location)$$

4.6 The SE Switching Module (SM)

The Switching Module (SM) is connected with all the input packet data and signaling ports of SE i.e., SEI-0 to SEI-3. An icon of SM, extracted from figure 4.4, is shown in Figure 4.7. The signal details of SM ports are described in Table 4.4. The SM

processes the packet transport function at each of its input port with the help of its in-house cascaded sub-switching modules (SSM). Each SSM does packet routing by interacting with the SE's Output-Port-Selector component through SM's output ports.

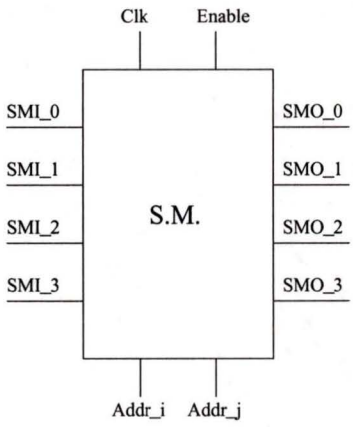


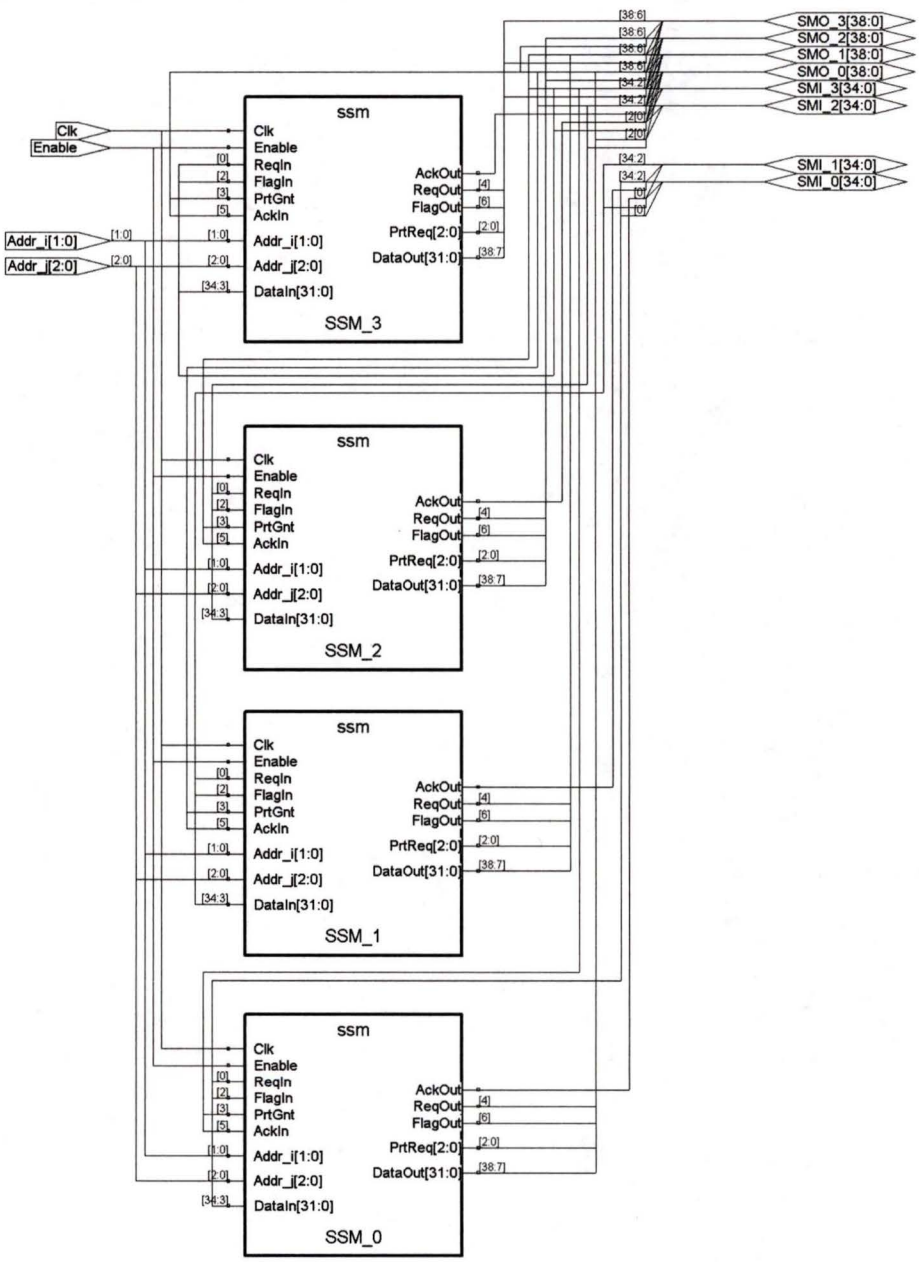
Figure 4.7. *An icon of SM*

Table 4.4. *The SM signal description*

Signal	Type	Description
Clk	In	Clock signal.
Enable	In	When asserted signifies valid address on input address lines, and ready signal for all internal FSMs.
Addr-i [1:0]	In	SE stage address within ILN.
Addr-j [2:0]	In	SE vertical address within a stage.
SMI [34:0]	InOut	SM input ports: 0 to 3, Each port contains in & out control signals, and input data [31:0] lines
SMO [38:0]	InOut	SM output ports: 0 to 3, Each port contains in & out control signals, and output data [31:0] lines

The internal structure of SM is shown in Figure 4.8. The figure shows the SM's input and output switching port-pipes are splitted and connected to the control and data ports of each SSM. Thus the hub of switching activity lies at the SSM, whose

c:\project files\switch_element\synthesis\rev_1\sm.srs 10/05/01 17:05:13
Instance SM: Sheet 1 of 1



Page: 1

Figure 4.8. *SM internal structure*

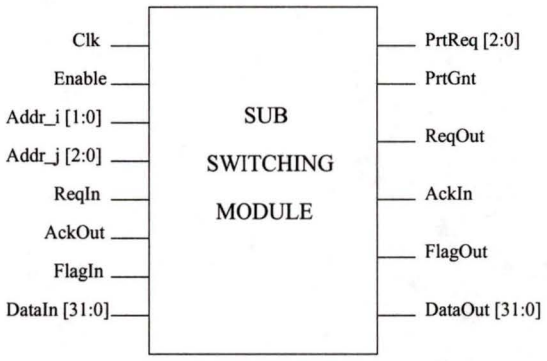


Figure 4.9. *The SSM*

signal description will also explain the SM's each port signal distribution. The detail of SSM is described in the next section.

4.7 The Sub Switching Module (SSM)

The Sub Switching Module (SSM) is the hub of switching functions in the SE. An icon of SSM is shown in Figure 4.9, and its signal description is detailed in Table 4.5. This module is modular and is replicated at each SM port, and subsequently on each input port of the SE. It functions independently to route the incoming packets, using its built-in Bit-wise Routing Algorithm (BRA).

The SSM activates on assertion of Enable signal, received from the Address Generator module of the SE. During its operation the SSM remains in three modes: ready, path establishment, and packet transport. When ready, the SSM keeps hunting the SE input port for a packet request. In path establishment phase, receives packet arrival indication, performs routing and establish desired destination path according to the routing information. This phase involves communication with preceding and succeeding SEs, along with the internal SE OPSel device. In packet transport phase, the SSM transports the actual packet data over its data lines.

The SSM is a processor whose applications are mainly decomposed into control and the functional units, as per definitions of stored program computer [55]. The control and functional units works by interacting with each other on a command and response principle. In this procedure, the control unit behaves as the master

Table 4.5. *The SSM signal description*

Signal	Type	Description
Clk	In	Clock input
Enable	In	Asynchronous initialization signal
Addr-i [1:0]	In	Address i - stage location
Addr-j [2:0]	In	Address j - vertical location
ReqIn	In	Request signal from preceding SE or switch IM indicating a packet arrival.
AckOut	Out	Asserted two times in a packet processing session, for the preceding equipment. First, to confirm system availability; Second, to indicate path availability status.
FlagIn	In	Indicates valid data over data input lines during destination address and packet transfer phases.
DataIn [31:0]	In	Packet data input lines; the data is picked from these lines at two instances - first for destination address, and second during actual packet transfer
PrtReq [2:0]	Out	Port Request signal to the OPSel. The one MSB acts as a control bit, whereas the two LSB bits indicates the required output port number.
PrtGnt	In	Port Grant - indication from the OPSel about the availability of desired port
ReqOut	Out	Same as ReqIn, but towards succeeding SE or switch OM.
AckIn	In	Same as AckOut, but from succeeding SE or switch OM.
FlagOut	Out	Same as FlagIn, but towards succeeding SE or switch OM.
DataOut [31:0]	Out	Same as DataIn but towards succeeding SE or switch OM.

and functional units as slave or decoders. The control unit in itself implements a Finite State Machine (FSM), which sequence the control commands output in a fixed pattern. Whereas, the functional units executes packet switching in accordance to the control commands received. In the design the functional units are five entities named as: Sensor, Translator, Router, Request, and Timer. The FSM communicates with the decoders over Command and Response buses. These buses in hardware consists of a bunch of wires, which are connected to the control unit and the individual ports of each decoders, respectively. The interconnection of the control and functional units is shown in Figure 4.10, and their individual functions are explained in the following sub-sections.

4.7.1 SSM Control Unit

The SSM Control Unit is composed of a FSM, which sequences the entire operation of SSM. This unit activates on assertion of Enable signal from the Address Generator module of the SE. Once activated, the FSM sequence the SSM switching function as: hunt for an incoming packet request at the input; on arrival of packet request send acknowledgment signal to the preceding equipment, indicating system's availability; perform path establishment process; on path establishment with the destination port, transport actual packet data; on completion of packet transport, release system's resources and go back in hunting mode for a new packet request.

The path establishment process involves receiving destination address, creating routing vector according to BRA, and seeking desired port from the OPSel of SE. In this process, if the desired port is not available, the system is responsible for providing either other port option or send negative acknowledgment to the preceding equipment.

The actual transport of packet is done once the path establishment is achieved after receiving and sending positive acknowledgment signals from the succeeding and to the preceding equipments, respectively. During this process the system has an option of implementing the network congestion control mechanism. This is done by observing the acknowledgment-in signal from the succeeding equipment, which is when removed, the data is suspended until the acknowledgment signal is received back. At this instance, the data transport is resumed again. In case, the congestion

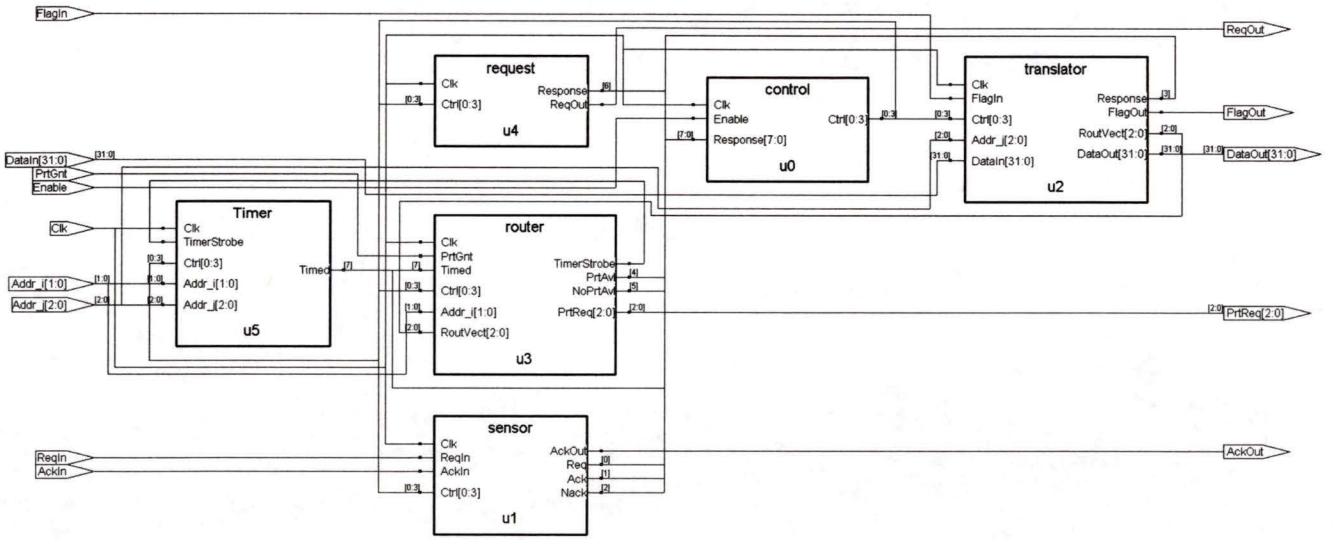


Figure 4.10. *The SSM structure*

remains for a longer duration, the system starts the mechanism of dropping the packet. During this whole process, the system keep interacting with the preceding equipment through acknowledgment-out signal.

The FSM interacts with the functional modules by sending control commands over the command bus and receives their responses over the response bus. These buses are practically a bunch of wires, and operates on one-hot coding principle. The slave modules listens to the appropriate command and react along with a response signal. The control FSM transit between eleven states during the entire operation of the SSM's switching function. The state diagram of the FSM is shown in Figure 4.11, and the operations executed in each state are explained below briefly.

In the Figure 4.11 the FSM is always sensitive to the input asynchronous Enable signal, which is when low it remains in 'IDLE state'. In this state, the FSM keep sending disable command to all the functional units of SSM. On assertion of Enable signal, the FSM moves into the 'READY state', and sends control command for a corresponding functional unit to hunt any packet arrival request signal. On receipt of packet request, being notified by the functional unit, the FSM moves to 'ACKNOWLEDGE state', and issues command for the functional unit to send positive acknowledgment signal to the preceding equipment, signifying equipments availability. After this action on expiry of timer, the FSM moves into 'TRANSLATION state', and sends command to receive destination address and create routing vector. On receipt of a response from the functional unit, the FSM moves to 'ROUTING state', and commands to access desired SE output port, according to the routing vector. The functional unit if responds with the desired port available signal, then FSM moves into 'REQUEST state', otherwise into 'NEGATIVE ACKNOWLEDGMENT state'. In REQUEST state, the FSM asserts control command to send a path establishment request to the succeeding equipment, and wait for their response. If the functional unit senses and responds with next equipments availability, the FSM moves to 'DESTINATION state', otherwise it moves back to the 'ROUTING state' for next available alternate routing routes. In DESTINATION state, the FSM commands to send destination address to the next equipment, and wait for their response. If the functional unit senses and responds an establishment of path with the ILN network destination port, the FSM moves into 'TRANSPORT state', otherwise it moves back to

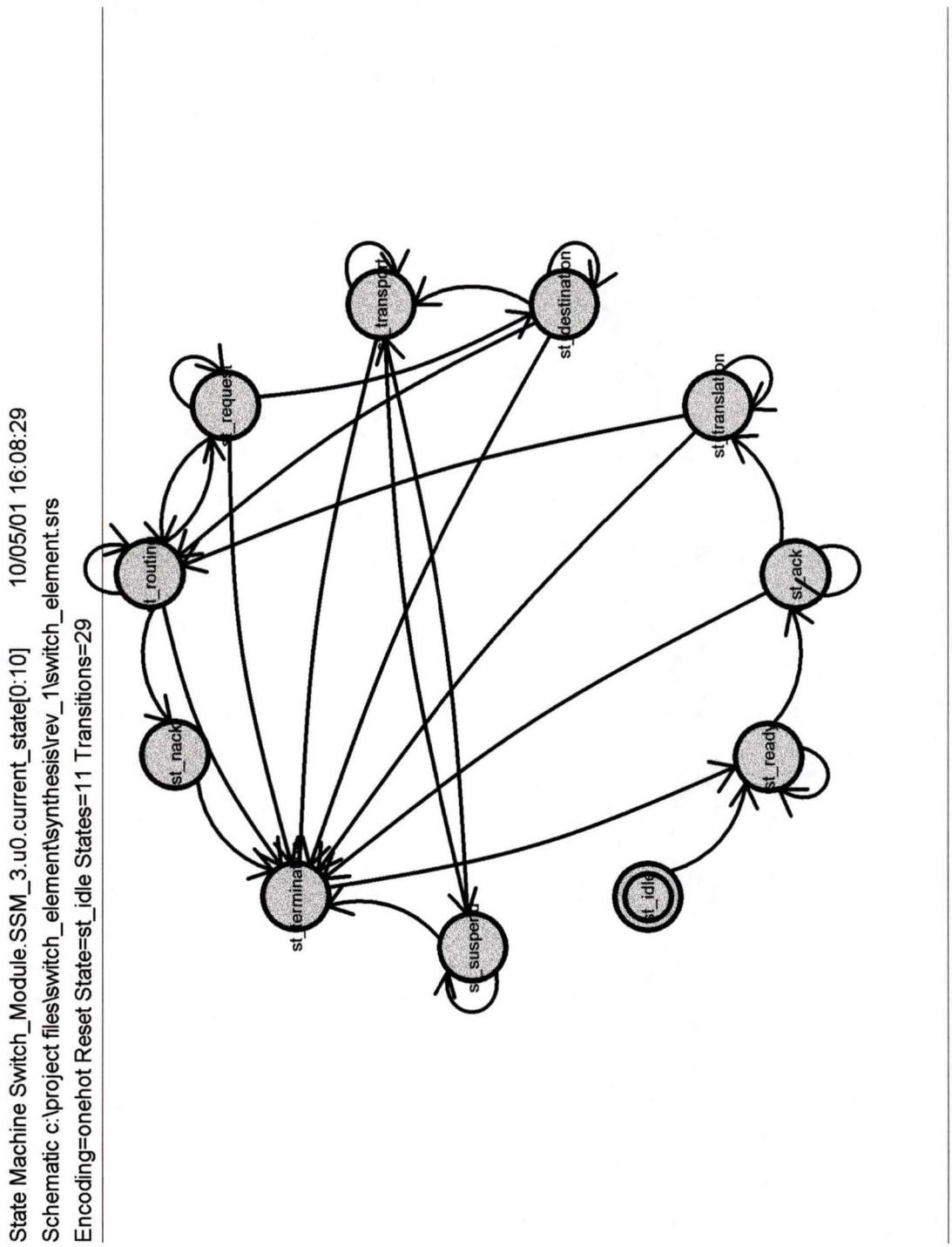


Figure 4.11. *SSM Control States*

the 'ROUTING state' for next available alternate routing routes. In 'TRANSPORT state' the FSM asserts command for the transport of actual packet data. During the packet data transport phase, if the decoder senses and reports a network congestion, the FSM moves into 'SUSPEND state', otherwise on end of packet data it moves into 'TERMINATION state'.

The NEGATIVE ACKNOWLEDGMENT state of the FSM is responsible of sending a command for the functional unit to assert NACK signal for preceding equipment informing an ILN network congestion from this SE. The preceding equipment may then attempts for path establishment through an alternate SE. In the SUSPEND state, the FSM issues command for suspending the transport of packet data along with network congestion report to the preceding equipment. The FSM moves back to the TRANSPORT state to resume packet data transport once the congestion is removed. However, in case of a long presence of congestion, the system has the option of terminating the session with necessary signaling to the preceding equipment of the SE. This feature ensures avoidance of stucking the system for ever in a single state. In the TERMINATION state, the FSM asserts command for all functional units to reset their registers at the reference position. The FSM in next clock cycle moves back to the READY state for serving next incoming packet request. At any clock cycle during the packet switching process, the FSM could move back to the READY state through TERMINATION state, whenever the input packet request signal is removed from the preceding equipment.

4.7.2 SSM Functional Units

Referring to Figure 4.10 the functional units of SSM in the design are: Sensor, Translator, Router, Request, and Timer modules. The task of these modules is to execute packet switching process after decoding the received commands from the Control unit. These modules are individually connected with the SSM control unit over command and response buses; along with, the input and output SSM ports control (Request, Acknowledge, Flag) and packet data (Data) lines, as shown in Figure 4.10. The functional modules communicate with preceding and succeeding SSMs, and with the SE's internal OPSel device over input and output control lines, to establish a path to the desired destination ILN network port. Further after path establishment, they

performs the function of actual transfer of the packet data over input and output ports data lines. The modules are further responsible to respond the outcome of its operations to the SSM Control unit. Each module returns its registers to their initialization values during the Idle state of SSM, and at every termination of packet switching session. A brief explanation of these modules is described as follows.

The functions of *sensor* are: sense packet request, sense positive or negative acknowledgment from the succeeding equipment, and send positive or negative acknowledgment to the preceding equipment. The function of *Request* module is to assert output request signal for the next equipment. The functions of *Timer* module are to insert various clock cycle delays in the switching process.

The functions of *Translator* module are: receive valid destination address when input flag signal is asserted, create routing vector according to the received destination address, send routing vector information to the Router functional module, send destination address along with corresponding output flag signal to the next equipment, receive packet data, and send or pause packet data along with corresponding output flag signal to the next equipment.

The functions of *Router* module are: receive routing vector information from the translator module, seek desired output port according to the routing information by sending specific port request to the OPSel, and seek alternate port when desired output port is not available. The Router, considers address i in routing since in the last stage of ILN network only straight connection is to be sought.

4.7.3 The SSM Mechanism

The SSM mechanism works on the interaction of SSM control and functional units to perform packet switching function at each port of the SE. A summary of the SSM control unit's commands and the functional unit's corresponding operational responses are explained in the Table 4.6.

In the Table 4.6 the control unit's 11-states are shown, and against each state the functional units functions: 'A' through 'M' are indicated and explained, respectively.

Table 4.6. *SSM control commands and Decoders corresponding functions*

Functional Unit Responses													Control Unit Commands
A	B	C	D	E	F	G	H	I	J	K	L	M	
-	-	-	-	-	-	-	-	-	-	-	-	-	IDLE
.	.	.	.	.	.	.	.	.	.	.	.	X	READY
.	.	.	.	.	.	.	.	.	X	.	.	X	ACKNOWLEDGE
.	.	.	.	.	.	.	X	.	.	.	.	X	TRANSLATION
X	.	.	.	X	.	.	.	.	.	.	.	X	ROUTING
.	X	.	X	.	.	.	.	.	.	X	.	X	REQUEST
.	.	X	X	.	.	X	.	.	.	.	X	X	DESTINATION
.	.	.	X	.	X	.	.	.	X	.	X	X	TRANSPORT
.	.	X	X	.	.	.	.	.	.	.	X	X	SUSPEND
.	.	.	.	.	.	.	.	X	.	.	.	.	NEG-ACKNOWLEDGE
*	*	*	*	*	*	*	*	*	*	*	*	*	TERMINATION

where,

'-' : Reset to initial values

'X' : Assert function

'*' : De-Assert function

'.' : Don't Care

wherein,

Function - A: Timer Module - insert OPSel Response Delay

Function - B: Timer Module - insert Next SE Response Delay

Function - C: Timer Module - insert ILN Destination Port Response Delay

Function - D: Request Module - send Request signal

Function - E: Router Module - seek SE Output Port

Function - F: Translator Module - send Packet Data

Function - G: Translator Module - send Destination Address

Function - H: Translator Module - create Routing Vector

Function - I: Sensor Module - send Negative Acknowledgment

Function - J: Sensor Module - send Positive Acknowledgment

Function - K: Sensor Module - sense Next SE Response

Function - L: Sensor Module - sense Path Establishment Acknowledgment

Function - M: Sensor Module - sense Packet Request

4.8 The SE Output Port Selector (OPSel)

The Output Port Selector (OPSel) block provides access to the desired output port of SE to the individual SSM modules. Another important function of the OPSel is to resolve the packet request contention over output ports sought by the input ports of SE. Each input of OPSel is connected with the outputs of individual SSMs; whereas, its outputs are connected with each output of the SE, respectively. The OPSel input port is a pipe of signals containing 35 wires of packet control and data signals, 3 wires of Port Request (PrtReq) signal by the SSM, and 1 wire of Port Grant (PrtGnt) signal from OPSel to the SSM. Each OPSel output port is a pipe of wires containing only 35 wires of packet control and data signals, which is consequently the output of the SE.

The OPSel performs its function by sampling the input packet request signals from each SSM, in a cyclic fashion. At a particular time slot if the module senses the corresponding SSM's packet request signal, it responds by a port grant signal to the SSM along with extending connection of 35 wires of packet control and data signals with the output port of SE. However, when a particular output port is busy, the module ignores any packet requests for that port; it is the responsibility of the requesting SSM to time out it's request signal.

A block diagram of OPSel structure is shown in Figure 4.12. The module comprises of 4 times cascaded internal buses, selectors, and a component identified in this design as Output-Port (OUTPORT) access device. All OPSel inputs are terminated over internal SSM buses: 0 to 3; and, all OPSel outputs are connected through OUTPORT component. The system samples the input SSM's packet request signal with the help of Selectors. Each selector is controlled by the respective OUTPORT component through selector control (Sel) signal. The OUTPORT sends Sel signal at every successesive clock cycle until a packet request for that particular OUTPORT is not received. In a certain clock cycle if a packet request matches to the OUTPORT address, the OUTPORT seizes sending Sel signal. This action connects the requesting SSM's 35-wire control and data signals with the output port of the SE. The OUTPORT thereafter sends port grant signal to the connected SSM for further processing of the switching function.

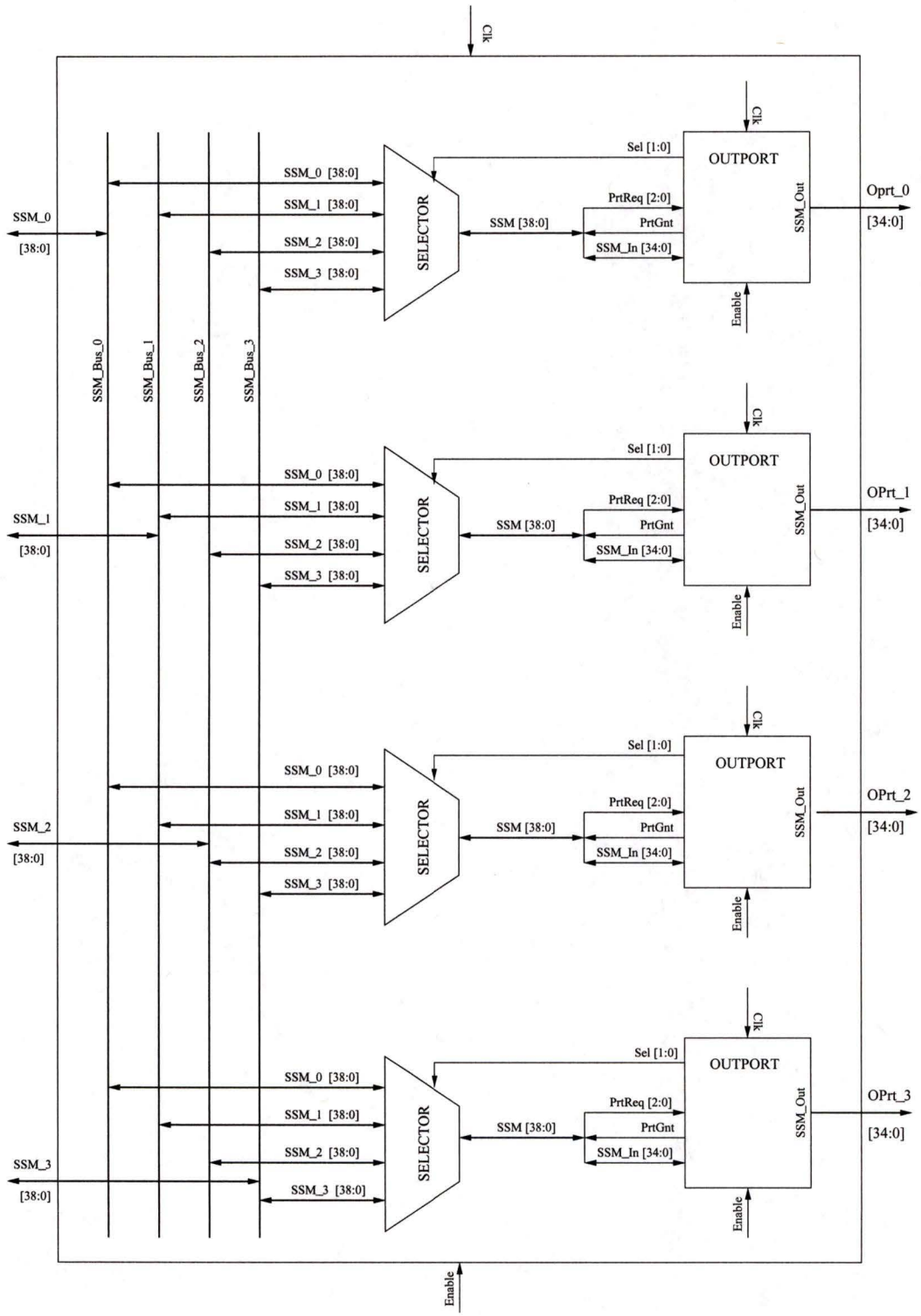


Figure 4.12. *The OPSel structure*

4.9 Summary

In this chapter we discussed the digital logic specifications for the hardware implementation of ILN network. These specifications were based on the ILN network theory, discussed in chapter 3. This chapter also discusses the methodology for devising the designs hardware implementation work, with definitions of industry standards.

The ILN network specifications gives details of the complete hierarchy of switch fabric. Each component is being specified with its functions and necessary hardware signals required to perform their intended functions. The heart of the ILN network control mechanism is also elaborated with its sequence of operation to initialize the switch fabric system, and perform packet switching processes.

The design is made simple for better understanding of ILN network theory and the VLSI implementations design steps. The circuit complexity is minimized by fixing the input and output ports of ILN network at 8; however, the design has the option to include this complexity to make the design scalable. Similarly, the complexity in Bitwise Routing Algorithm is also made straight forward, by not including the intelligent routing techniques. The switch fabric is still efficient without these complexities in the circuit, the design has the capability of resolving contention, is fault tolerant and fully depicts the complete characteristics of the ILN network switching theory.

Chapter 5

Hardware Design of ILN Network

5.1 Introduction

In this chapter the ILN network hardware implementation is described, being based on the design specifications drawn in the previous chapter. The chapter discusses hardware design flow in detail, which entails writing design specifications in hardware description language, performing model's functional simulations, synthesis, and physical implementation.

The simulations of the design are done using the ModelSim simulation tools from Model Technology Incorporated (MTI), being a trade mark of Mentor Graphics Corporation [58]. The synthesis is done using Synplicity's SynplifyPro-7.0.2 tool [59]. The physical implementation of the design is achieved using Xilinx Integrated Synthesis Environment (ISE) tool [60]. The target technology for the physical implementation of ILN network is Xilinx Virtex-II [61] Field Programmable Gate Array (FPGA) [62, 63] integrated circuit (IC) chip.

This chapter is organized on down-to-top approach of ILN network design hierarchy, in order to understand the chain sequence of functions within the network. *Section* 5.2 provides a detail of ILN network hardware implementation's design flow along with the review of applied technology definitions. *Section* 5.3 defines the signaling conventions applied for control signaling between various design modules. *Section* 5.4 shows the results of behavioral simulation of SE functions. This section details all possible functional aspects of SE for its activation and during switching processes. The functions of SE are individually articulated in separate sub-sections and shows simulation results for addressing, switching, packet flow control, fault tolerance, alternate routing, and contention resolution. *Section* 5.5 shows results of SE synthesis

and physical implementation in the FPGA. *Section* 5.6 shows results for ILN network, being builded by interconnecting the constructed SEs. This section includes ILN network simulation results depicting a complete cycle of routing an incoming packet from an input to the output port of the network. The section further exhibits the results of ILN network synthesis and its FPGA implementation. The *Section* 5.7 of the chapter presents hardware optimization techniques, being studied in literature and experienced during the design process of ILN network. In the end, the chapter concludes the ILN network hardware implementation work.

5.2 The ILN Network Hardware Design Flow

The ILN network hardware design flow is based on generic FPGA design techniques, being generally followed by a designer in the industry [64]. The design flow achieves target implementation work in three broad phases. These phases can also be assumed as domains focusing on different design aspects: *function*, *structure* and *geometry* [28, 29].

The *function* domain of the design flow presents an abstract description of a system's function. The entire system is described in terms of algorithms and conventional computer programming language. This level of functional modeling is often called 'behavioral modeling', and is achieved during the simulation process of the design. In this phase the EDA tool creates a logic, which is technology independent. The logic represents data storage registers by using variables, and transformations by arithmetic and logical operators.

The *structure* domain of the design flow describes the system in terms of units of data storage registers and transformation. This phase is achieved during Synthesis process of the design. This operation creates a logic which is technology dependent. The created logic depicts system's operation using a data path and control section. The data path contains data storage registers where data is transferred between them through transformation units. The control section sequences operation of the data path components. The logic represents generated circuit in terms of wires and interconnected electronic components.

The *geometric* domain of the design flow describes a system in terms of VLSI

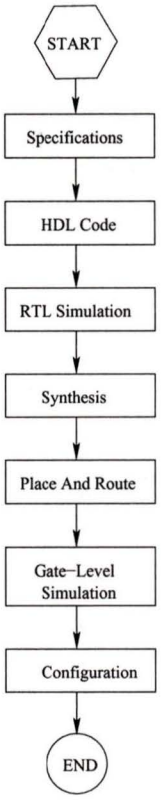


Figure 5.1. *Hardware Design Levels of Abstraction*

circuit implementation using a floor plan. This final phase of design flow is achieved during the Place and Route (PAR) process of the design.

The ILN network FPGA design flow, adopted in this thesis work is shown in Figure 5.1. The input to the hardware implementation work is the ILN network's specifications; being already drawned in the previous chapter of this thesis. The following are the definitions and explanation of each level of design flow abstractions.

Hardware Description Language (HDL) Code:

The method of specifying a design to EDA tools as input is either done by drawing a schematic, or by specifying Boolean expressions, or by writing a Hardware Description Language (HDL) code [69]. The industry's popular EDA tools prefer HDL design entry methods. In this design flow the method of ILN network specifications input to the EDA tools is done by writing a HDL code. The HDL looks much like conventional high-level computer programming languages. However, conventional

programming language tends to the concept of executing a single statement of a program at a time; which is unsuitable in hardware environment. Since, the hardware operation is inherently parallel i.e., all gates are constantly sampling their inputs and computing new outputs; thus, the concept in HDL programming is purely based on high degree of parallelism. Due to this reason, the HDL code writing approach has to be different than the conventional programming language code. In the industry there are three known HDLs: ABEL, VHDL, and Verilog. The ILN network design is being implemented in VHDL environment.

VHDL is a language for describing digital electronic system designs [65, 66]. The language was developed by the USA government Defense department's Very High Speed Integrated Circuits (VHSIC) program. In the course of this program, a standard language for describing the structure and function of integrated circuits (IC) had been evolved as VHSIC Hardware Description Language (VHDL). It was subsequently developed further under the auspices of the Institute of Electrical and Electronic Engineers (IEEE) and adopted in the form of IEEE Standard 1076 as *standard VHDL Language Reference Manual* in 1987. VHDL fulfills a number of needs in the design process. First, it allows description of the structure of a system i.e., how it is decomposed into subsystems and how those subsystems are interconnected. Second, it allows specific description of the functions of a system using typical programming language forms. Third, it provide means for simulation and test using test-benches. A VHDL testbench is just another entity that instantiates the created design, and provides stimulus test vectors to be applied on the inputs of the design for verifying that the correct outputs are generated by the design under test. Fourth, it allows the detailed structure of a design to be synthesized from a more abstract specification to a specific target technology.

RTL Simulation:

The Register Transfer Level (RTL) is an electronic circuit transformation of a designs specification [58]. The EDA tool creates and optimizes the logic, while compiling, using technology-independent operations. The technology independent operation uses software provided characteristics of the logic, instead of actual component's behavior. The logic thus created by RTL depicts functional performance of model in accordance to the applied algorithms of HDL.

The simulation process verifies the operation of a design before it is being implemented in hardware. The RTL simulation allows to observe the circuit behavior at the model's input, output and internal nodes. This process verifies the HDL syntax and determine if the functions of the circuit is according to the specifications. The VHDL testbenches are used to specify circuit input stimuli and output responses; which are further used to test design specifications at various points in the design flow. The design at this point is device independent, therefore no timing information is available at this stage.

Most design development is usually done through iterative RTL simulations until the final functionality is achieved.

Synthesis:

Synthesis is the process of constructing a gate-level netlist from a model of a circuit described in HDL code [59]. In the Synthesis process the EDA tool generates a logic by translation and optimization, while compiling, using technology-dependent operations. The tool further maps the generated logic into the the target technology and creates a netlist for the Place and Route processing of the design. The process consists of multiple stages of translation and optimization.

The translation process performs transforming a level into another i.e., HDL to RTL, to Logic, and to Gate level. In the process the EDA tool uses technology specific library components and the user defined constraints. The constraints are defined by user in a text file and contains the timing and layout information, which affects the logical design implementation in the target device. The optimization methodology targets to optimize for area first, and then the timing delay. It takes a design through four main internal levels of intermediate refinement: RTL level, Logic level, Gate level and Mapping to target technology to produce Gate level netlist. At each of these levels the process of translation and optimization repeats before moving to next level until a netlist - binary file is produced.

In the RTL level process, the EDA tool checks the HDL syntax specific to synthesis process. A synthesis specific HDL syntax is slightly different than behavioral code. The system translates HDL code into a RTL level netlist. The generated logic is the graphical representation of hardware structure, according to the HDL specifics. The structure consists of blocks of logic represented by boolean equations. The op-

timization at this level involves scheduling, resource binding, datapath structuring, partitioning, and pipelining of the translated logic.

In the Logic level process the system translates the RTL netlist to register-transfer level blocks such as flip-flops, arithmetic-logic-units, and multiplexers interconnected by wires. The optimization process keeps register elements fixed, and only restructures the combinational logic. Further the boolean optimization includes minimization, equation flattening, and equation factorization.

In the Gate level process, the system translates and maps the logic netlist in target technology. This involves acquiring predefined components from a library of user-defined target technology. The target technology of this thesis design is Xilinx Virtex-II FPGA family, with XC2V250 device. The system extracts area and timing information from the cells of the targeted technology. Gate level optimization involves a process of looking at a local area of logic containing a few cells and trying to replace them by other cells from the technology library that fits the constraints better.

The map process first performs a logical Design Rule Check (DRC) on the design in the gate level netlist; and then maps the logic to the components (logic cells, I/O cells, and other components) in the target Xilinx Virtex-II FPGA. The mapped design file thus produced is used for the Place and Route of the design.

Place and Route:

The Place and Route (PAR) is the task of placing modules adjacent to each other to minimize area or cycle time; and, routing the modules by connecting with wires [60]. The PAR tool takes mapped design file as input from the synthesizer tool for the PAR of design into the target technology.

The PAR tool place and route each module independently. The modules are developed in parallel, so the post-routing timing results of one module can be used with placement and routing constraints of another synthesized module. In the Xilinx Development System, PAR places and routes a design using a combination of two methods: Cost-based PAR, and Timing-driven PAR. Cost-based placement and routing are performed using various cost tables which assign weighted values to relevant factors such as constraints, length of connection and available routing resources. Timing-driven PAR places and routes a design based upon the user-defined timing constraints.

The tool, after PAR process, generates an annotated Standard Delay Format (SDF) file to be used for the Timing Simulation and a fully routed Native Circuit Description (NCD) file to be used by the the Implementation tool for the configuration of FPGA.

Gate Level Simulation:

The Gate Level Simulation is performed after synthesis and Place and Route processes, using back-annotation information [58, 60]. The back annotation processes generate a netlist of library components annotated in a Standard Delay Format (SDF) file. The gate level simulation uses timing information from the SDF file, being based on the delays in the placed and routed design. Thus this simulation is also called as the Timing Simulation.

Timing simulation describes the circuit behavior far more accurately than functional simulation. It includes detailed timing information for the targeted device, and verifies that the design runs at the desired speed for the user-defined constraints. This process verifies timing relationships and determines the critical path signal delays for the design under worst case conditions. It can also determine whether the design contains setup or hold violations.

The procedures for functional and timing simulations are nearly identical, and requires a HDL testbench for input stimulus to run the simulation.

Configuration:

The Implementation tool processes to convert the logical design represented in the design source into a physical file format that can be implemented in the selected target device [60]. In Xilinx ISE tool the implementation processor is named as Bit Generator (BitGen). The BitGen creates a Configuration file, which is used to program the target FPGA so that it can execute the desired function.

The BitGen takes a fully routed Native Circuit Description (NCD) file, from the PAR process, as its input and produces a configuration binary file. This binary file is a bit stream file which contains design information for internal logic and interconnections of the FPGA, plus device specific information from other files associated with the target device. The binary data in the BIT file can then be downloaded into the FPGA's memory cells, or it can be used to create a PROM file, e.g. Flash memory of a system. At the power-up, the FPGA configures its internal logic by using the BIT

file from the system memory.

The Hardware Design Flow, discussed above, is typically repeated to try targeting various devices to determine the most suitable fit to the design specifications. If a particular device proves to be too large or too slow for the needs, a designer selects a smaller or faster target device from same or a different FPGA family. In each domain, the design flow returns to HDL level, and the designer repeats the whole process, until the desired and satisfactory results are accomplished.

5.3 ILN Network Signaling Conventions

The ILN network path establishment phase requires hand-shaking mode signaling in between the engaged preceding and succeeding equipments. Referring to Figure 4.10, the control signaling is done over *Request*, *Acknowledgment*, and *Flag* wires of each ILN network port. Since the ILN network ports are attached with SEs, and within ILN network the SEs are interconnected with each other, therefore the same principle of control signaling remains valid for the SEs also.

According to the SSM Control module's FSM states (refer Figure 4.11), the SE system requires to send and receive positive acknowledgment (ACK), negative acknowledgment (NACK), and valid data present flag (FLAG) signals to the preceding and succeeding equipments, during packet switching processes. The function for asserting ACK and NACK signals is performed by the Sensor sub-module of the SE's SSM; while the function of asserting FLAG signal is performed by the Translator sub-module of the SE's SSM. The signaling conventions used in this system design are defined and explained below.

Sending Positive acknowledgment (ACK):

The SE is required to send positive acknowledgment (ACK) signal to the preceding equipment at two instances. First, when path establishment phase is initiated; and Second, when path is established with the destination ILN network port. This signaling involves input commands (Ctrl), input request (ReqIn), and output acknowledgment (AckOut) wires of Sensor functional module.

In the first case, while the system in ready state, when the SSM senses a packet arrival signal over ReqIn control wire, it sends back an ACK signal to the preceding

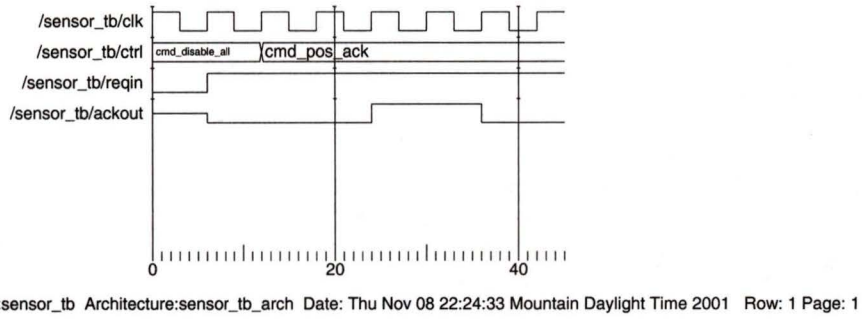


Figure 5.2. *Positive Acknowledgment Signal*

equipment over AckOut wire, signifying SE's availability. This signal in the design is a continuous two clock cycle high potential, and is shown in Figure 5.2. In the Figure, on assertion of ReqIn signal, the system responds over AckOut wire with a '1' for two clock cycles, and then returns back to '0'.

An equivalent logic circuit to achieve this function is shown in Figure 5.3. In the Figure, when the module decodes command for an appropriate action it triggers the Value register to send a '1' to the register One in the first clock cycle, and a '0' in the next clock cycle. This sequence travels towards register Two. In the circuit the XOR gate keep sending a '1' AckOut until it continues sensing a difference of potentials over registers One and Two. As soon this difference is removed, the XOR returns AckOut to the '0'.

In the second case, when path is established, the ACK signal is a continuous '1', as long as there exists no congestion in the network. In this case, the system simply passes on the ACK coming from next equipment to the preceding equipment. Thus, the functional module does not needs a complex circuit for this function.

Sending Negative Acknowledgment (NACK):

The SE requires to send negative acknowledgment (NACK) to the preceding equipment when there exist no output port available for the desired destination. This signal is a one clock cycle high potential. This function is performed by the SSM Sensor sub-module, which when senses appropriate control command for sending NACK signal, it raises a one clock cycle high potential over the SE's 'acknowledgment out' (AckOut) port. A demonstration of this function is shown in Figure 5.4 wherein the sensor sub-module when receives appropriate control command, it asserts '1' over

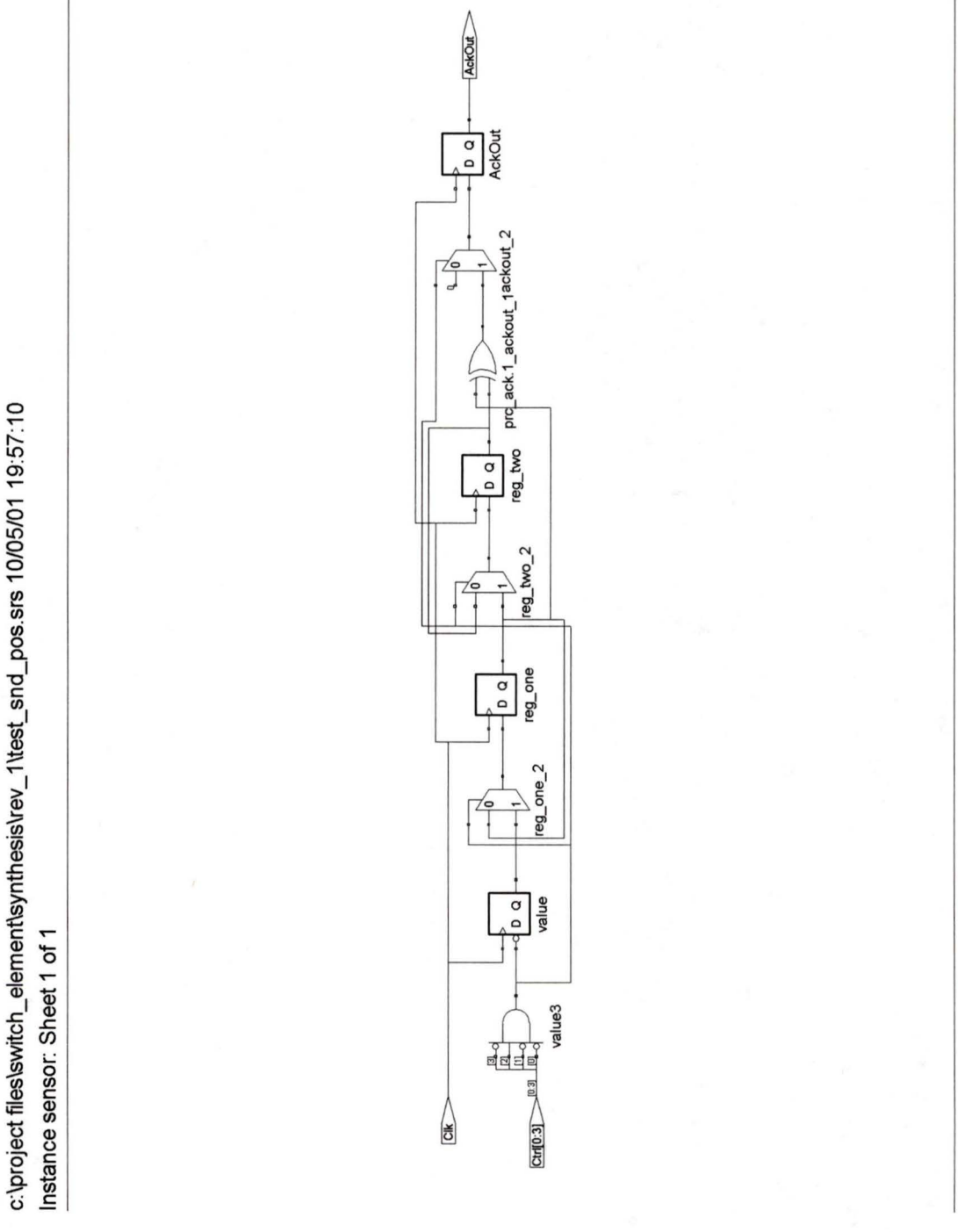


Figure 5.3. *Positive Acknowledgment Logic*

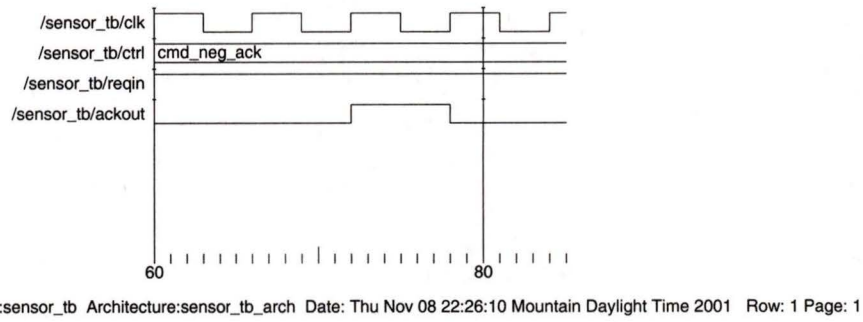


Figure 5.4. *Negative Acknowledgment Signal*

AckOut port for one clock cycle, and returns to '0' in the next clock cycle. An equivalent logic circuit to perform this function is shown in Figure 5.5. The logic is similar to the positive acknowledgment signal sending function; except that instead of XOR an AND gate is used to combine the register one and two outputs. In the circuit, the AND gate has an inverter at one of its input, to know the previous clock cycle state of the first register. The sensor sends a '1' output over the AckOut port for only one clock cycle, after decoding the send negative acknowledgment control command at its input.

Sensing ACK and NACK Signals:

The ILN network system also needs to sense and decode the incoming positive or negative acknowledgment signals from the next equipment. This function is achieved in SE by sensor sub-module. The acknowledgment signals reach at the input of sensor through 'Acknowledgment In' (AckIn) port of the SE. A demonstration of these functions is shown in Figure 5.6. The module on receiving an appropriate control command, remains into the mode of wait until an acknowledgment signal is received from the next equipment. The sensor component sends its response to the SE's control module over positive acknowledgment (ACK) and Negative Acknowledgment (NACK) ports and wires, accordingly. In the figure, when at AckIn an ACK signal of a '1' for two clock cycles arrives, the module sends a '1' to the Control unit over ACK wire. Similarly, when at AckIn a NACK signal of a '1' for one clock cycle arrives, the module sends a '1' to the Control unit over NACK wire.

An equivalent logic to perform these functions are shown in Figures 5.7 & 5.8. In the figure 5.7, the incoming acknowledgment signal is passed through two Registers:

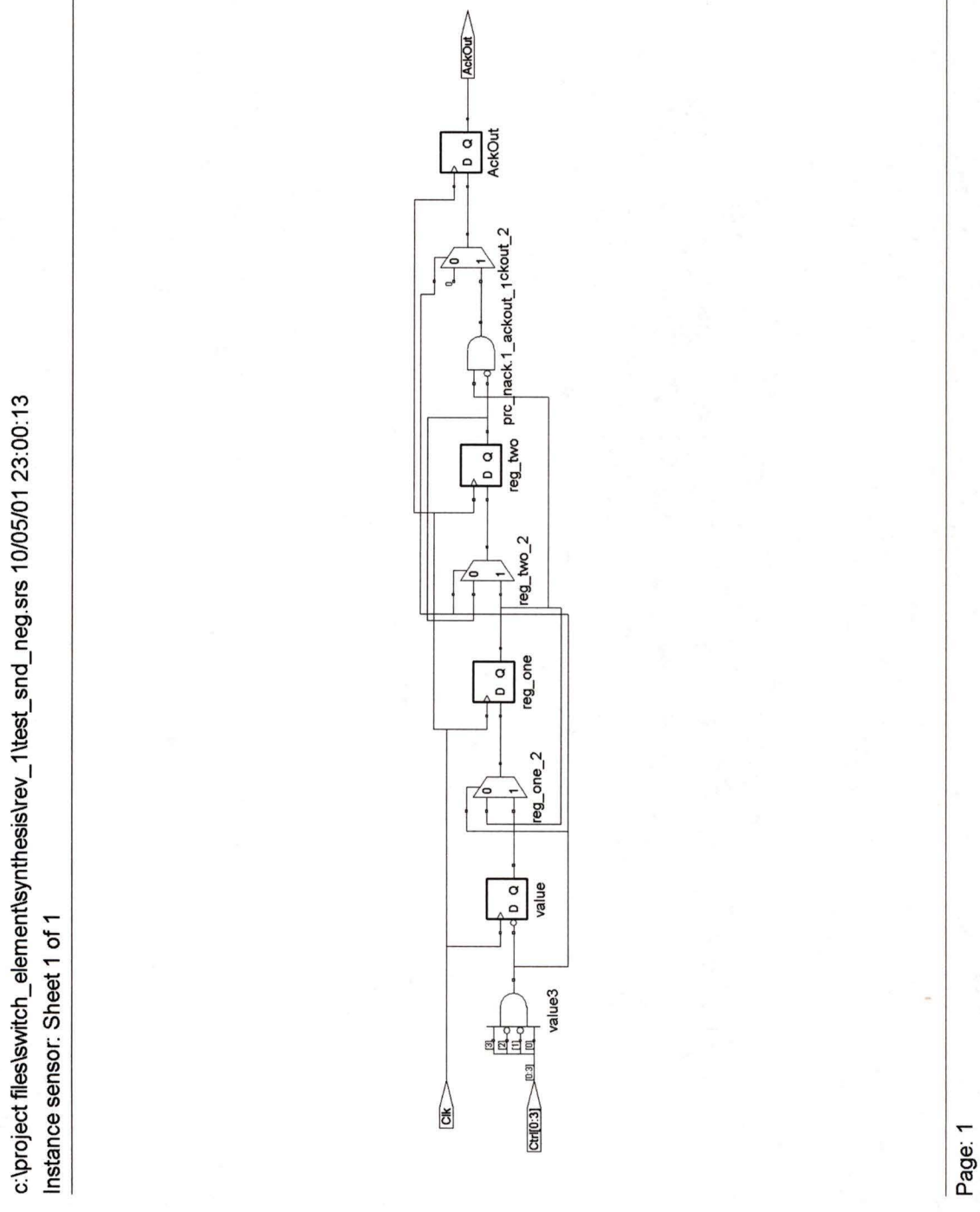


Figure 5.5. *Negative Acknowledgment Logic*

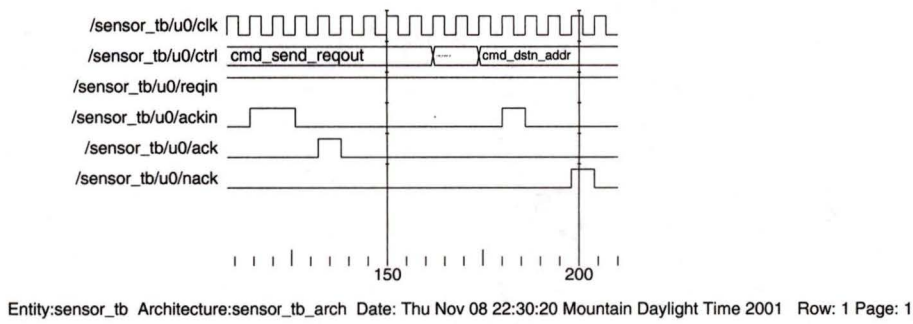


Figure 5.6. *Sensing Positive/Negative Acknowledgment*

One and Two. An AND gate when finds '1' on both registers, i.e., a continuous '1' for two consecutive clock cycles, it sends a high potential signal over ACK output port, signifying a positive acknowledgment has been received at the input (AckIn) of the Sensor module. In other condition it keeps the ACK port at low potential. This ACK port is connected by wires to the Control module through response bus at the SE's structural layer.

However, in figure 5.8 the function of sensing negative acknowledgment over incoming acknowledgment signal is achieved with a little complex logic. The initial part of the circuit is similar to the function of sensing positive acknowledgment. But, the output of the circuit (NACK) is derived with the help of a selector after analyzing input control command and the status of both registers - One and Two. The state analysis of registers One and Two is achieved by employing three AND gates and an inverter. If in two consecutive clock cycles both registers are at '1' (ACK signal input), then the AND and inverter circuit resets the values of registers to '0'. The circuit mechanism only asserts the Nack output port to high potential when register Two is at '1' and the register One is at '0', signifying a negative acknowledgment has been received at the input (AckIn) of the Sensor module. The Nack output port is connected by wires to the Control module through response bus at the SE's structural layer.

Valid Data Signal:

The control signal *FLAG* is used to signify a valid data over data lines. This wire is pulled up for both incoming and outgoing packet data transmission.

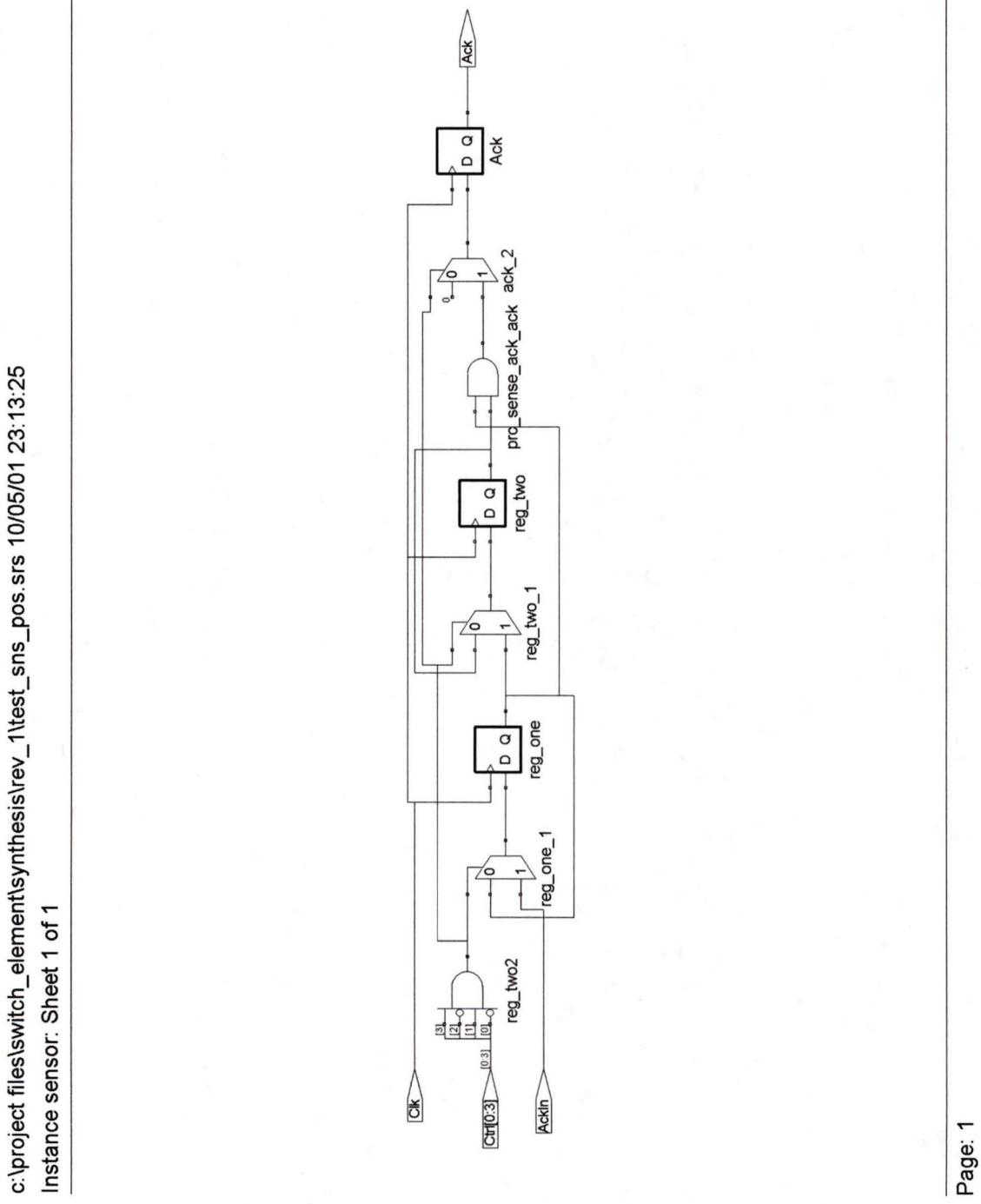


Figure 5.7. *Sensing Positive Acknowledgment Logic*

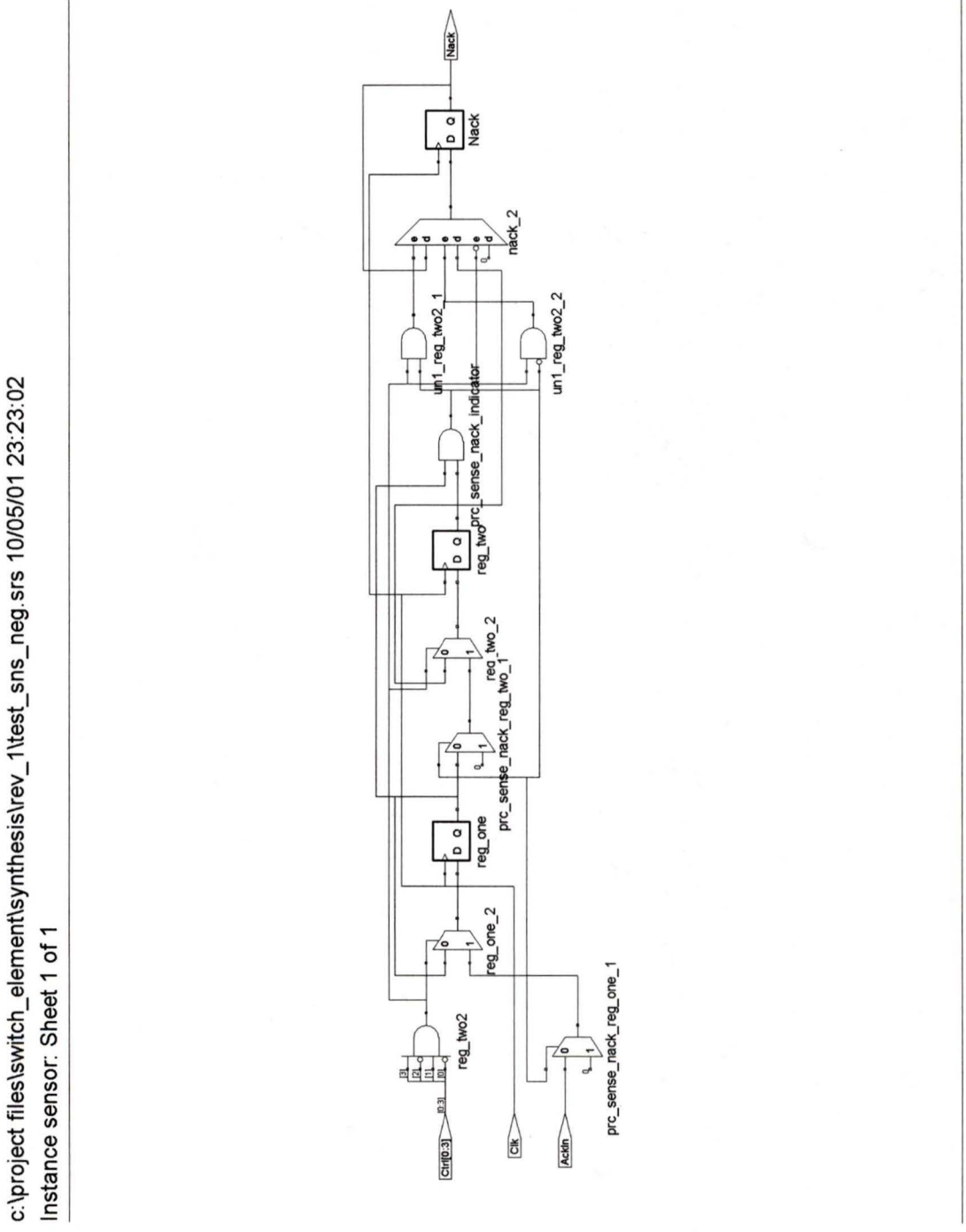


Figure 5.8. *Sensing Negative Acknowledgment Logic*

5.4 The SE Simulation Results

The SE VHDL design behavior is tested and verified in simulations, applying test benches for the model devised in Figure 4.4 and signals described in Table 4.2. A complete simulation cycle of SE's initialization and packet switching process, according to the specifications, is shown in Figure 5.9. The figure shows four SE switching input ports: SEI-0 to SEI-3, four output switching ports: SEO-0 to SEO-3. Each port has three control signal wires: *request* (Req), *acknowledge* (Ack), and *flag* (Flag); and one 32-bit data signal wire. The SEI's are connected with the preceding equipment to the SE; its 'Ack' wire sends signals in outward direction whereas, its other wires receives signals in inward direction. The SEO's are connected with the succeeding equipment of the SE; its 'Ack' wire receives signals in inward direction whereas, its other wires sends signals in outward direction. The SE functions are sensitive to its input Reset signal, which when remains high, the SE performs its initialization function at onset, and thereafter continue switching processes.

5.4.1 SE Addressing Function

Referring to the simulation cycle demonstration shown in Figure 5.9, the system's registers take values at the rising edge of the clock signal. The period of clock (Clk) signal is calibrated at 5 nano-seconds (ns), considering a frequency of 200 MHz. The SE is presumed to be at 0th location of ILN network, since its 'Sign' port is tied to a '1' potential. Thus, the SE's addressing function generates its addresses i and j as "00" and "000", respectively. Further, the addressing function generates output addressing information (for next SEs) over its output address ports. In the figure, the output address ports are providing the control information at the 0th bit of addr-i, as per specifications. The SE addressing system further generates 'Enable' signal after the specified delay, at 600 nano-seconds (ns).

5.4.2 SE Switching Function

As per ILN network design specifications, the SE's switching functions are performed at it's SSMs. Once the SSM receives Enable signal assertion, its control FSM goes into Ready state, to receive and process arriving packet requests. During the packet

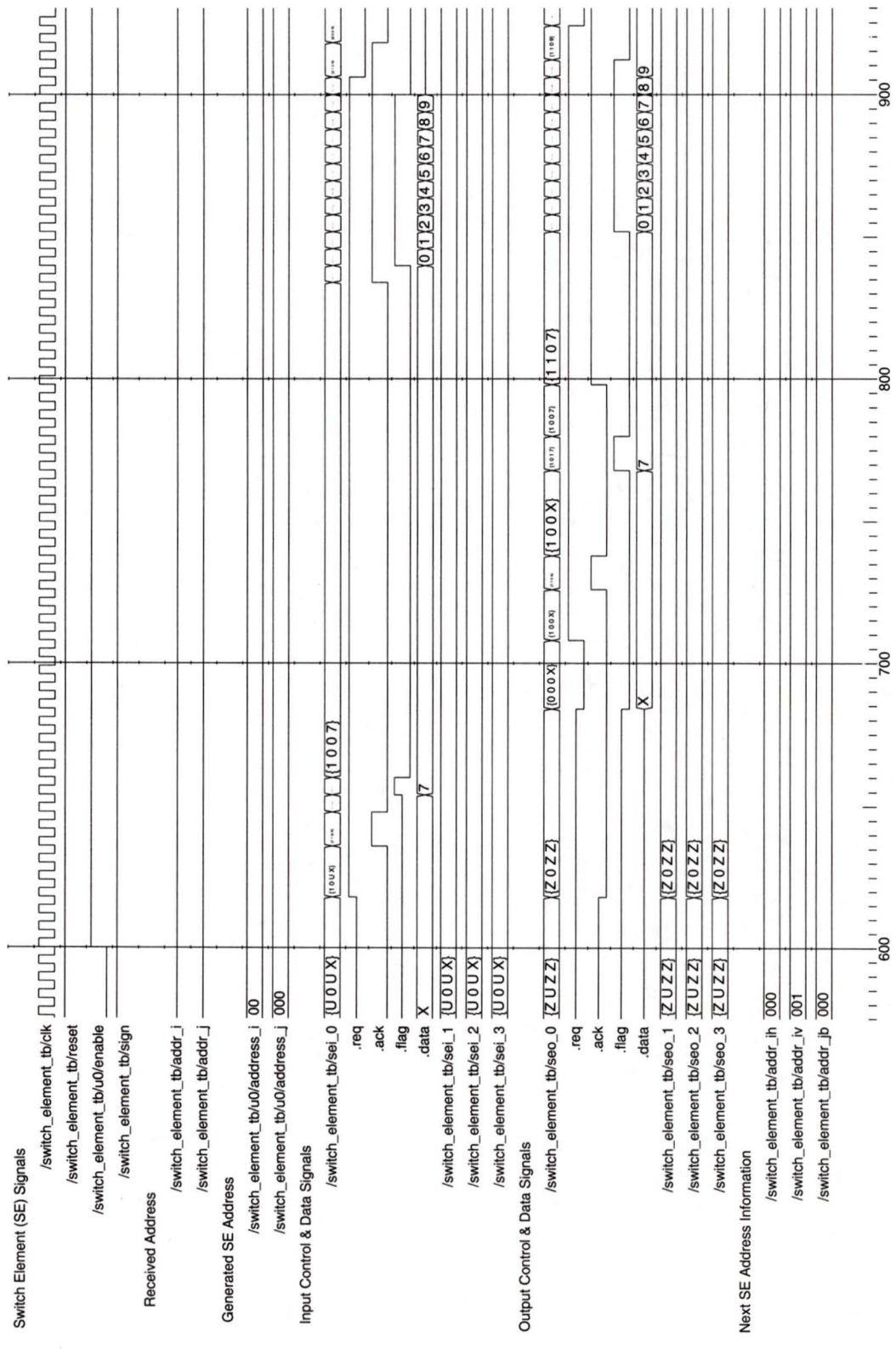


Figure 5.9. *A complete packet switching cycle*

processing phase the SE always remain sensitive to its input Request signal wire.

In the figure, a packet request from the SE's preceding equipment is received on SEI-0 at 615 ns; the Sensor module of SE generates and sends back an Acknowledgment (ACK) signal, signifying the SE is healthy and available. At 655 ns, the SE receives a destination address (for ILN network port '7') along with the FLAG signal from the preceding equipment. The SE performs translation function and creates a routing vector (RV), according to the Bit-wise Routing Algorithm (BRA). In accordance to the RV, the SE sends a Request signal at its Cube-0 output port (SEO-0), for the next SE connected with SEO-0 within ILN network. The SE receives ACK signal at 725 ns from next SE, signifying a healthy and available succeeding equipment. At 770 ns the SE sends the required destination address along with the assertion of FLAG signal to the next SE. At 795 ns, the SE receives second ACK signal, signifying availability of path for the desired destination. The SE, at 830 ns, sends back ACK signal to its preceding SE, informing the desired path has been established. At 840 ns, the SE receives packet contents at its Data wires along with the corresponding FLAG signal assertion. The SE at 850 ns starts transporting the received packet contents for the next SE over its output Data wires of SE0-0, along with corresponding FLAG signal assertion. The SE, during the state of packet transport phase replicates the received flag and data signals over its corresponding output wires; and, remains sensitive to both input Request and Acknowledge signal wires.

5.4.3 SE Packet Flow Control Mechanism

The SE design is capable of implementing packet data flow control mechanism, in response to network congestion situation. This function is accomplished during packet data transport phase of ILN network. The flow control mechanism of SE keep monitoring the input value at the ACK signal wire. In case the potential on this wire turns low, the control FSM of the system goes into the state of suspending the data, with appropriate de-assertion of ACK signal to the preceding equipment.

A demonstration of SE's packet flow control function is shown in Figure 5.10. The figure is also showing the FSM control command sequence. At 875 ns the input ACk signal is de-asserted by the next SE, signifying congestion in the network. The SE control system change its state and issues relevant command from "SEND DATA"

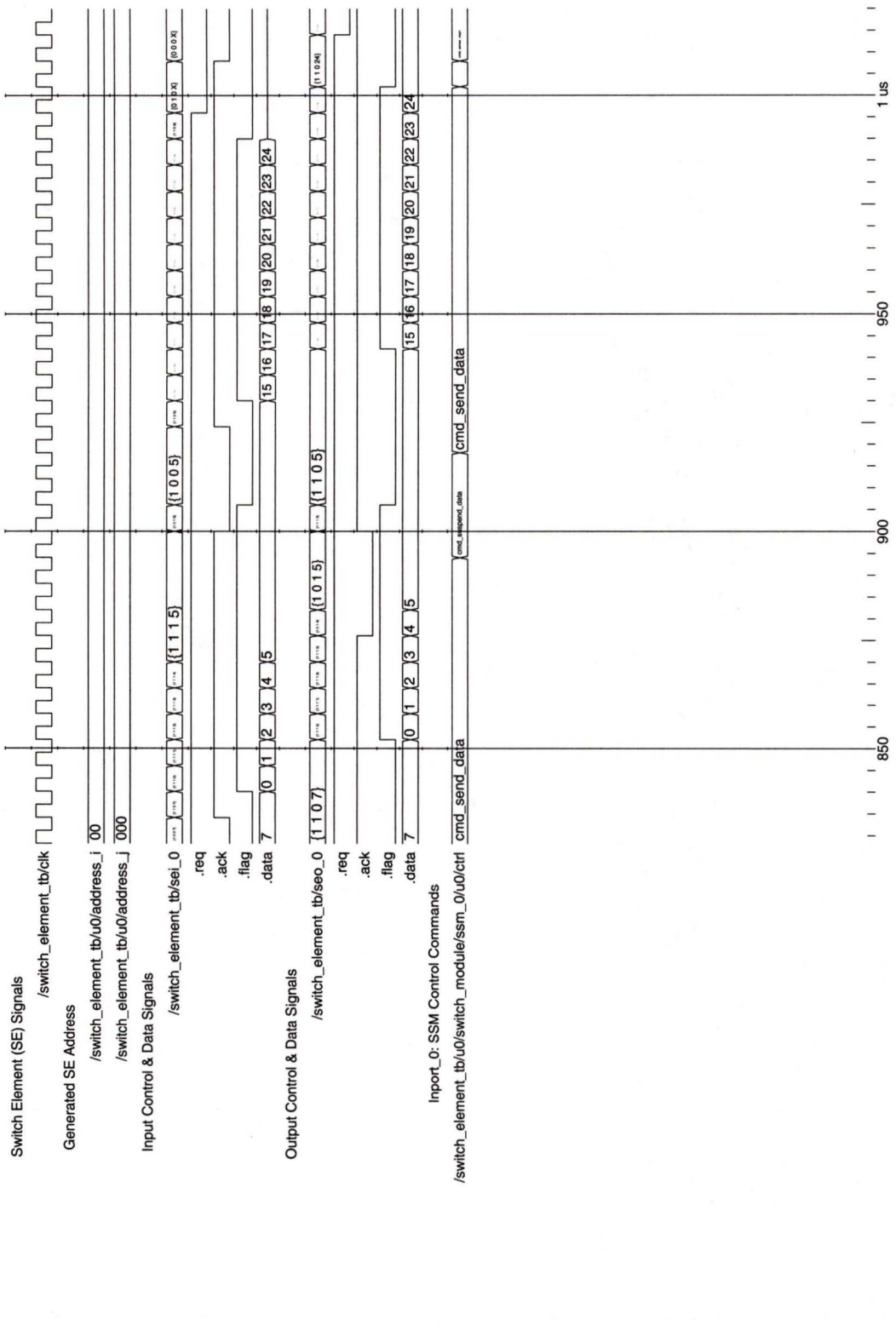


Figure 5.10. *Packet Flow Control Mechanism of SE*

to "SUSPEND DATA" for its Data Transport module function, along with necessary output ACK signal to the preceding equipment. The preceding equipment on absence of valid ACK signal removes the FLAG signal, which is being replicated by the SE on its output for the succeeding equipment. The system remains in the state of data suspension until again the input ACK signal is asserted by the next SE. In the figure at 900 ns the ACK input signal is again asserted by the next equipment, thus the SE resumes its function of packet data transport. The system also has the complexity of waiting for a certain number of clock cycles in the state of data suspension, to monitor longer congestion delays, with the help of its Timer functional unit. On expiration of timer delay the control may move for terminating the packet transport session.

The input value at Request signal wire, during packet data transport state, signifies end or continuation of a packet session. In Figure 5.9, at 905 ns, the input Request signal is being removed by the preceding equipment. At this, the SE prepares for the termination of packet session, and removes Request and Acknowledge signals from its output, and resets its registers of packet switching logic. After this process, the SE goes back into the Ready state to process a new incoming packet request.

5.4.4 SE Fault Tolerance and Alternate Routing Function

The ILN network is inherently fault tolerant, and thus SE has the capability to hunt up to $n+1$ available routes, to establish a path towards destination ILN network port. In case, the desired next equipment is not available (due to faulty link or faulty next SE), the SE attempts to access another ILN network node in accordance to the RV. However, if no path is available the SE's system sends a negative acknowledgment (NACK) signal to its preceding equipment; and goes back into ready state to serve another packet request. This function of SE is demonstrated in Figure 5.11. The SE after receiving packet request signal and required destination address at its input SEI-0 port, attempts to access next SE according to the generated RV. The SE successively sends and removes request signal on all of its output SEO ports, after expiration of its timer delay period in each event of no response from the succeeding equipment. After no response from its last SEO port (being straight connection within ILN network), the SE system sends a NACK signal back to its preceding equipment, and goes back into the ready state.

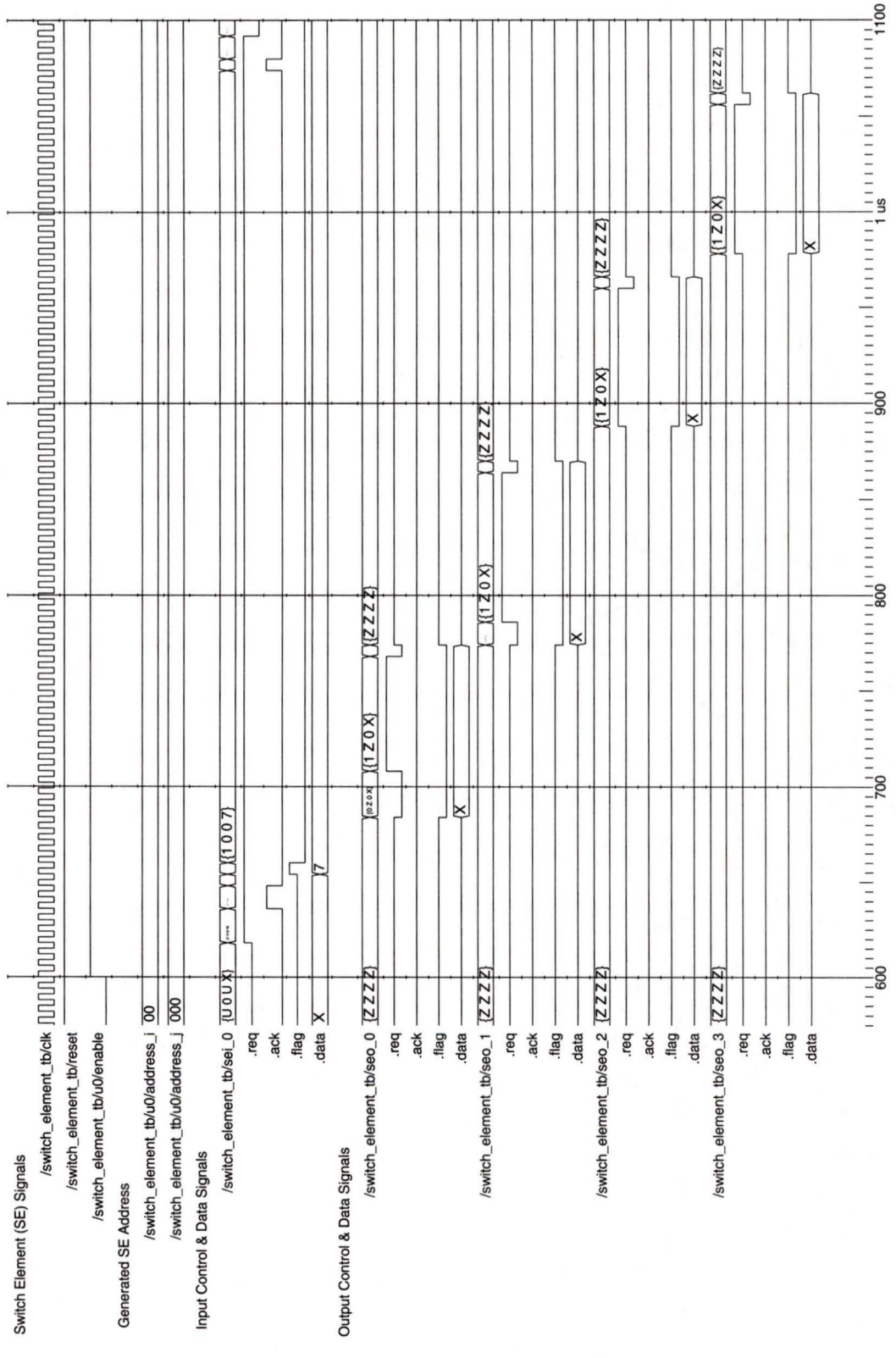


Figure 5.11. *SE alternate routes attempt*

An another scenario of a failed path establishment could be the availability of next equipment, but a congestion in the successive ILN network node, thereby receipt of NACK signal from the next SE. The SE function then reacts to look for other alternate routes, and repeat the process for path establishment. In case there exists no alternate path then the SE system sends NACK signal to its preceding equipment, resets its registers of switch processing logic, and goes back into the ready state to receive and process new incoming packet requests. This function of SE is demonstrated in Figures 5.12 & 5.13. In the figures, the SSM Control module generates command signals according to the FSM sequence shown in last chapter's figure 4.11. In this example, the SE receives packet request and destination address at its SEI-0 input port. The SE thereby sends request signals to the next SEs through its cube and straight connection output ports. The SE receives ACK signal only on its straight connection port; and thus it sends destination address along with the corresponding FLAG signal to the next available SE. Within the timer expiration period of SE, it receives NACK signal from the next SE, signifying a failure of successive path establishment. Since, now the SE has no more available ports to access it sends back NACK signal to its preceding equipment, and goes back into the ready state.

5.4.5 SE Contention Resolution Function

The SE has the capability to resolve contention among its input ports seeking same destination output port. This contention resolution mechanism resides in the Output-Port-Selector (OPSel) module of the SE. Referring to Figure 4.4 of last chapter, the OPSel's four output ports are individually connected with each output port of SE. Each output port, within OPSel, is driven by an OutPort module. Each OutPort module, i.e., each SE's output port, when idle, keeps hunting in cyclic sequence for a port request signal, from any input port of the SE. The SSM (an input port of SE), sends port request on a 3-bit port-request (PrtReq) signal wire, whose bit-3 is a control wire. The OPSel responds to the SSM port request by a port-grant (PrtGnt) signal wire. In case the output port is already serving a request it will not respond, the SSM is responsible for removing the port request after a certain delay. A demonstration of this feature of SE is shown in Figure 5.14. The figure is displaying two SE input ports (SEI-0 & SEI-1) are contending for a same output port

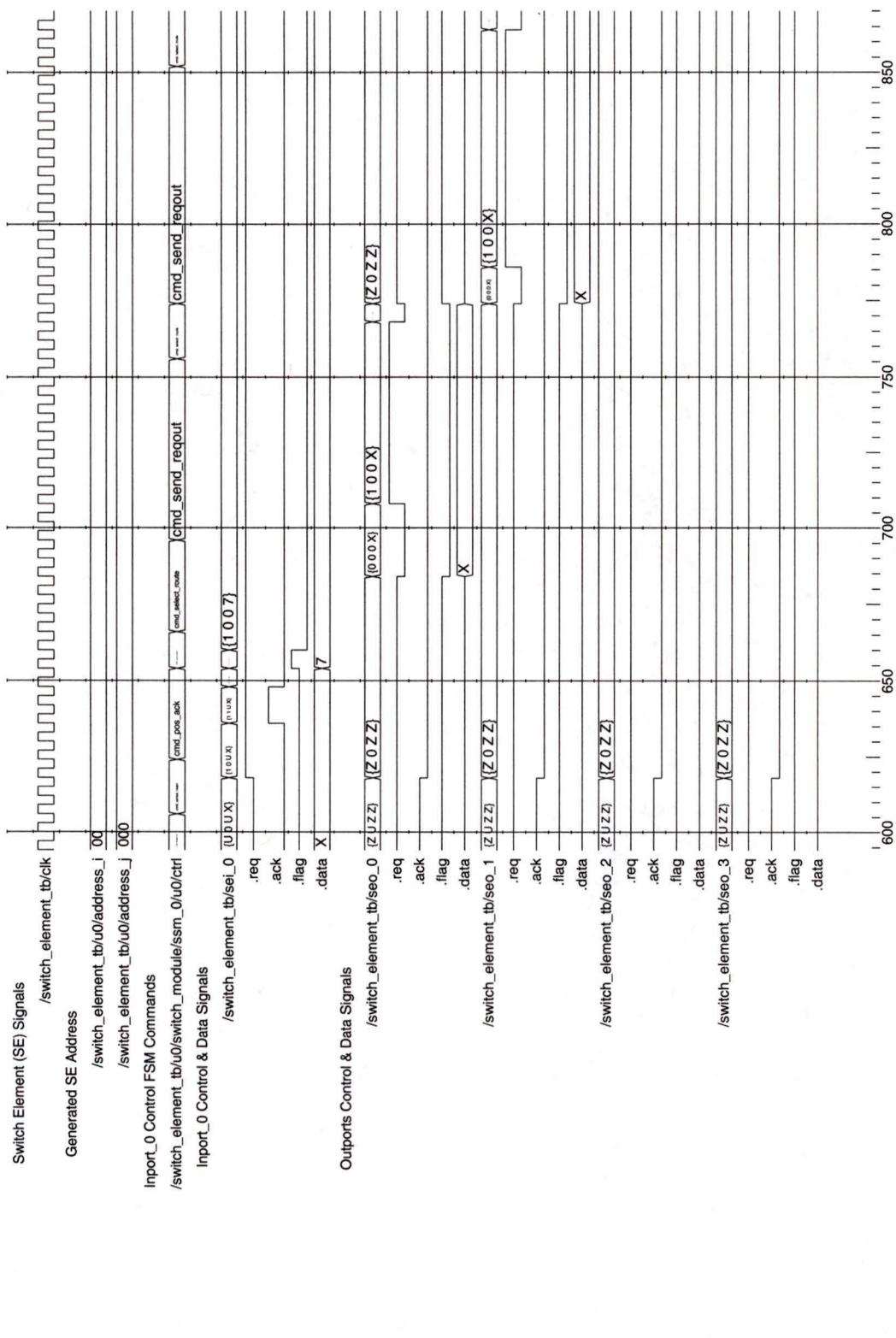


Figure 5.12. *SE response over NACK receipt - I*

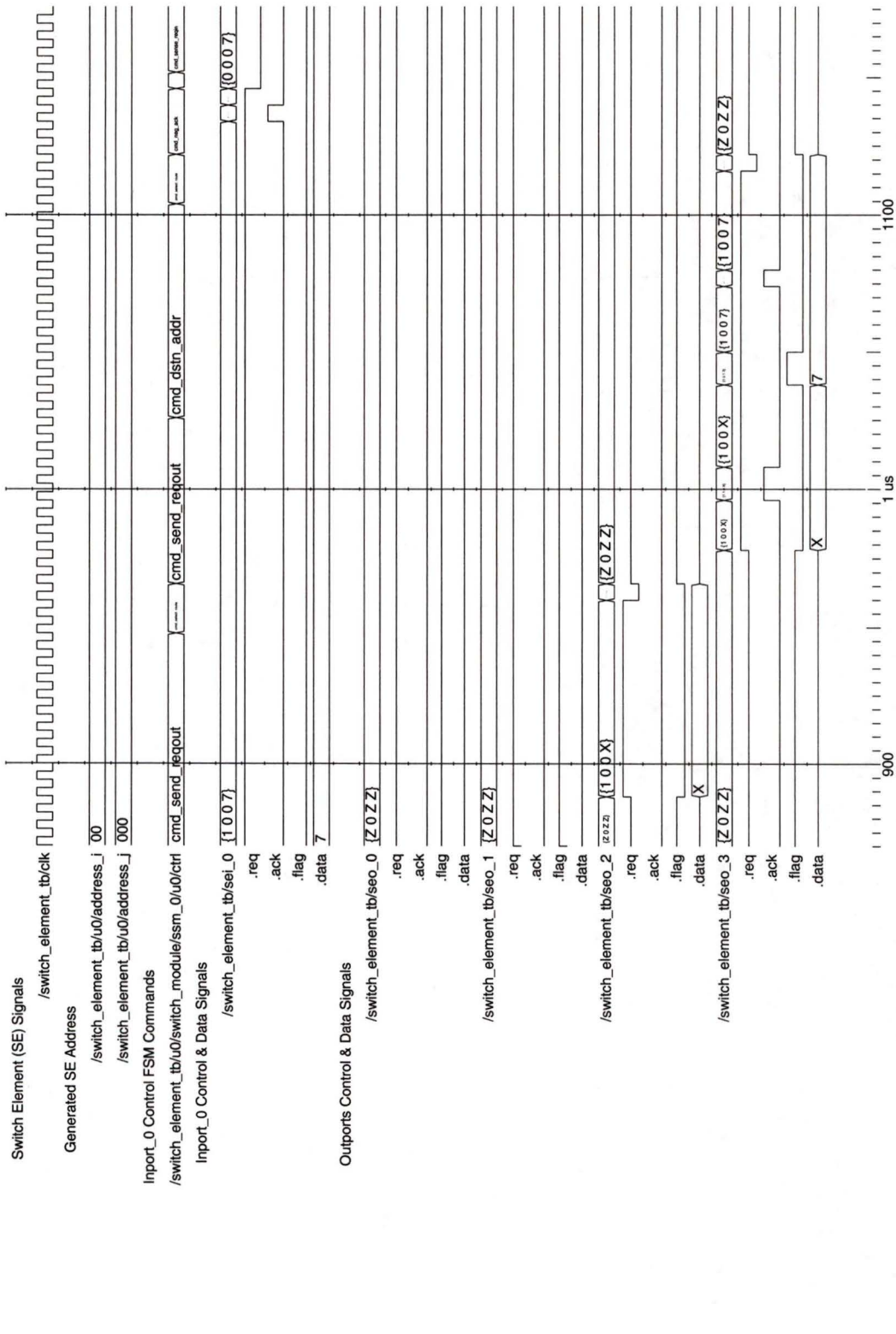


Figure 5.13. *SE response over NACK receipt - II*

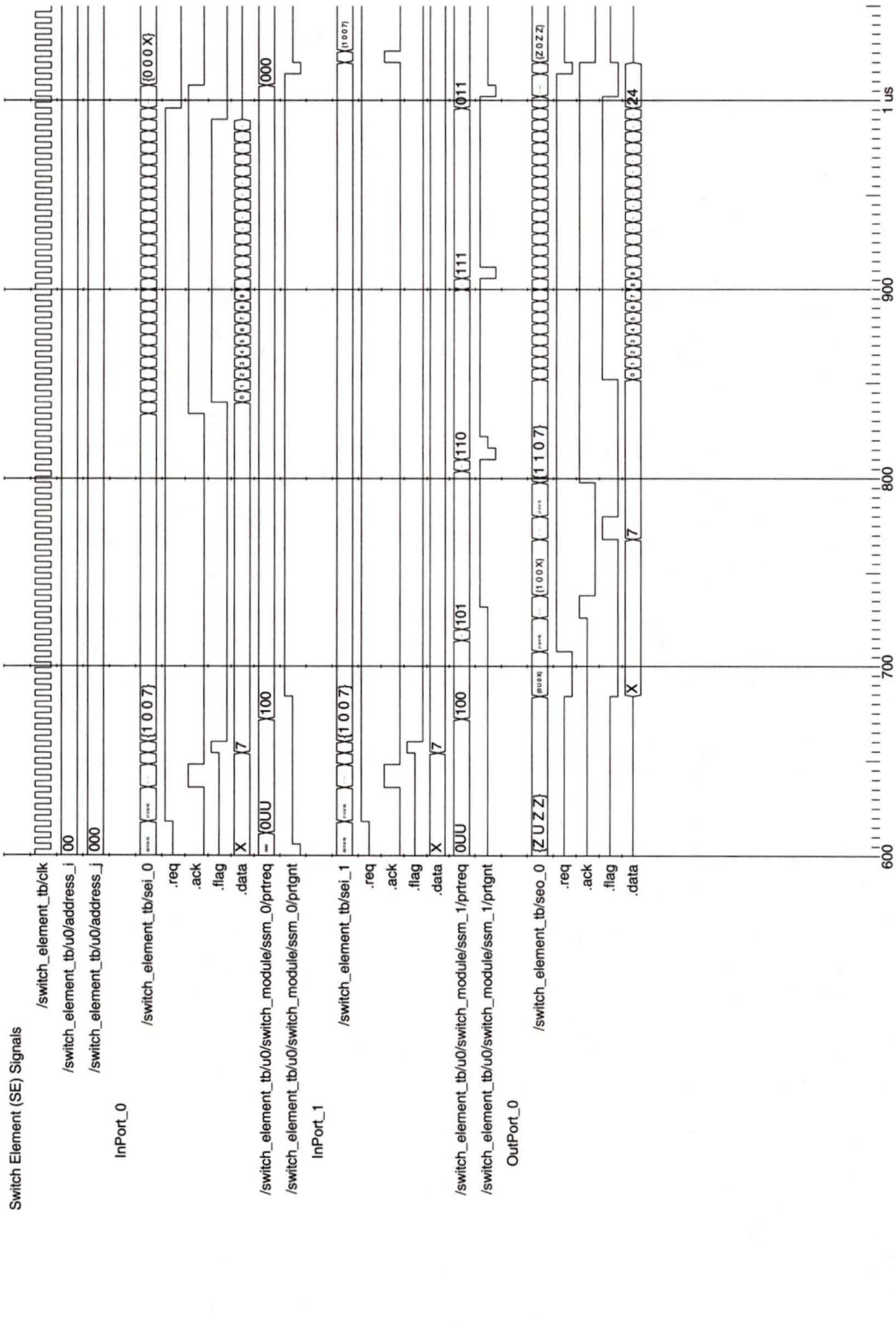


Figure 5.14. *The SE Contention Resolution*

(SEO-0). The figure is also showing the respective PrtReq and PrtGnt signals of both contending input ports. Two packets arrives at SEI-0 AND SEI-1, at the same time with same destination address requests. Both input ports sends a port request with their control bit pulled to '1', and output port address "00" to the OPSel of SE. The OPSel resolves this contention and sends port grant signal to the SEI-0; and thus this packet's request is further transmitted to the next SE over SEO-0, and is eventually transported to the destination address. However; the SEI-1, on absence of port grant, after a delay removes its previous request and sends successive port requests. The SEI-1 being successively receives port grant for other output ports, but due to the congestion within ILN network or non-availability of the destination port could not get the ACK signal from the desired port. This eventually results in sending a NACK signal to the SE's preceding equipment at SEI-1 input port.

In the above demonstration of various packet switching processing scenarios, it is also shown that the SE is fault tolerant. It is seen that in case of a NO response from the next equipment on SE's ports, it hunts forward for an alternate route. This process continues until all its resources are consumed. With the availability of redundant path establishment resources the SE is bound to access a routing path to the desired destination ILN network output port.

5.5 The SE Synthesis Results

The target technology for synthesizing the SE hardware design is Xilinx Virtex-II family, being mapped to part: XC2V250FG256-6. In the part number the XC2V250 is the device name, FG256 is the package, and -6 signifies the speed of the device. The start and the end point of paths in the logic are clocked by the rising edge of global clock signal.

A timing report of the design is shown in Table 5.1. The results show an estimated frequency of 156.2 MHz, giving 4.99 Gbps of data rate for 32-bit data lines. The report further exhibits a +0.5 slack. The slack is a parameter to gauge the timing performance of the design. Positive slack time values (greater than or equal to 0 ns) are good, while negative slack time values (less than 0 ns) indicate the design has failed timing requirements. The negative slack value indicates the amount by which

Table 5.1. *The SE Timing Report*

Clock	Requested Frequency	Estimated Frequency	Requested Period	Estimated Period	Slack
Clk	144.0 MHz	156.2 MHz	6.9 Nano-Sec.	6.4 Nano-Sec.	+0.5

the timing is off because of delays in the critical paths of the design. An observation of Timing Report further indicates results of worst case signal Delay on a critical path as 6.4 ns. The average delay is calculated as 6.3 ns, with set-up requirement of 0.1 ns.

A summary of FPGA resource usage report for the SE synthesized logic design is reproduced below:

Cell usage:

FDE .... 290 used

FD .... 63 used

FDC .... 56 used

FDP .... 1 used

GND .... 1 used

VCC .... 1 used

MUXF5 .... 165 used

FDR .... 194 used

FDSE .... 1 used

FDS .... 7 used

FDRS .... 1 used

XORCY .... 91 used

MUXCY-L .... 86 used

I/O primitives:

OBUF-F-24 .... 148 used

OBUF .... 2 used

IBUF .... 148 used

BUFGP .... 1 used

I/O Register bits .... 0 used

Register bits not including I/Os .... 613 used (19%)
Internal tri-state buffer usage summary:
BUFT + BUFE: 1 of 1536 (0%)
Global buffer usage summary:
BUFGs + BUFGPs: 1 of 8 (12%)
Mapping Summary:
Total LUTs: 1167 (37%)
where,
BUFE: Internal 3-State Buffers with Active High Enable
BUFGP: Primary Global Buffer for Driving Clocks
BUFG: Global Clock Buffer
BUFT: Internal 3-State Buffers with Active-Low Enable
FD: D Flip-Flop
FDC: D Flip-Flop with Asynchronous Clear
FDE: D Flip-Flop with Clock Enable
FDP: D Flip-Flop with Asynchronous Preset
FDR: D Flip-Flop with Synchronous Reset
FDRS: D Flip-Flop with Synchronous Reset and Set
FDS: D Flip-Flop with Synchronous Set
FDSE: D Flip-Flop with Clock Enable and Synchronous Set
GND: Ground-Connection Signal Tag
IBUF: Single-Input-Buffer
LUT: Look Up Table
MUXCY-L: 2-to-1 Multiplexer for Carry Logic with Local Output
MUXF5: 2-to-1 LookUp Table Multiplexer with General Output
OBUF: Single-Output-Buffer
OBUF-F-24: Multiple-Output-Buffers:24
VCC: Positive Voltage
XORCY: XOR for Carry Logic with General Output

From the above FPGA resource usage report it is deduced that the SE design is utilizing 1167 LUTs, being 37% resources of the target XC2V250 FPGA device. The

device has the total capacity of 250K system gates distributed in 3072 LUTs. The SE design resource requirement is calculated as 92.5K system gates, which are within the limits of the target FPGA device.

It is the standard recommendation that when the design synthesis timing result is within 5-10 percent of the desired result, then the place and route process be done to see if the implemented logic meets the specifications goal. The synthesis result shows that the SE design implementation is +7.8% over the set goals of 144 MHz frequency and 6.9 ns of clock period, which is within limits. Further, the '+' slack time result indicates that the timing constraints have been met. The PAR of SE thus implemented and the timing and hardware implemented logic's functionality were verified in the Gate Level Simulations. This result provides good confidence in the bit-file created at the end of implementation process. This bit-file is thus valid for the configuration of the target FPGA device.

5.6 The ILN Network Hardware Implementation

The ILN network design of this thesis is an 8-input × 8-output ports network. The switch fabric is made up of interconnected SEs. Each input and output port of ILN network is connected with an individual SE. According to Figure 4.1, the SEs connected with the input ports of ILN network utilize only their SEI-0 ports; whereas, SEs connected with the output ports of ILN network utilizes only their SEO-3 ports; the other ports of these SEs are left open. The Sign input port of 0th SE of ILN network is tied with a '1' potential, whereas the rest of SEs are kept at '0' potential. The ILN network's Reset input is connected with the 0th SE; however, the other SEs of ILN network receive Reset signal from their preceding SEs once valid addressing information is available on the respective SE's Address output ports. The ILN network operate with reference to the rising edge of a synchronous clock signal, received from the Switch system to the network. The functions and hardware implementation results of ILN network, according to the specifications, for packet switching are demonstrated and described in the following sub-sections.

5.6.1 ILN Network Simulation Results

The ILN network design for an eight port fabric requires a maximum delay of 120 clock cycles after the assertion of Reset signal for initializing all its internal SEs. It is assumed that the switch system will be responsible for this delay requirement in its design specifications. The functions of ILN network SEs involved in a complete cycle of packet switching are demonstrated and discussed below.

Consider a packet's request signal arrives at the 0th port of ILN network from the IM of switch. This ILN network port is essentially connected with the 0th SE. A demonstration of ILN network's SE-0 function in this scenario is shown in Figure 5.15. In the figure, the Sign bit of SE is pulled up to '1', since it is the 0th SE of ILN network. Thus, the SE's addressing function generates its addresses i & j as "00" and "000", during the ILN network initialization phase. During the packet switching phase, the SE receives a packet request at its input port SEI-0; and then the destination address for ILN network output port-7 (i, j: 11,111). The SE then performs path establishment function and sends packet request signal for the next SE of ILN network over its output port SEO-0, in accordance to the created RV using BRA. Refer figure 4.1, in the ILN network the SEO-0 of SE(00,000) is connected with the SE-9 (i, j: 01,001), being the next SE in this example.

The initialization and switching processes at ILN network's SE-9 (i, j: 01,001) are shown in Figure 5.16. In the figure, the SE Sign port is tied with '0' potential, since it is not the ILN network's 0th SE. Refer figure 4.1, the SE-9 receives input addressing information from SE-8; thus during the initialization phase, the SE receives preceding SE's addresses i & j as "011" & "000" over its input address lines. The bit-0 of input address i is '1', signifying the preceding SE is at the vertical location of the SE. Therefore, the SE's addressing function generates its addresses i & j as "01" and "001". During the packet switching phase, the SE when receives packet request and required destination address, it sends path establishment signals to the next SE over its output port SEO-1, in accordance to the created RV. Refer figure 4.1, within ILN network the SEO-1 of SE(01,001) is connected with the SE-19 (i, j: 10,011), being the next SE in this example.

The initialization and switching processes at ILN network's SE-19 (i, j: 10,011) are shown in Figure 5.17. In the figure, the SE Sign port is tied with '0' potential,

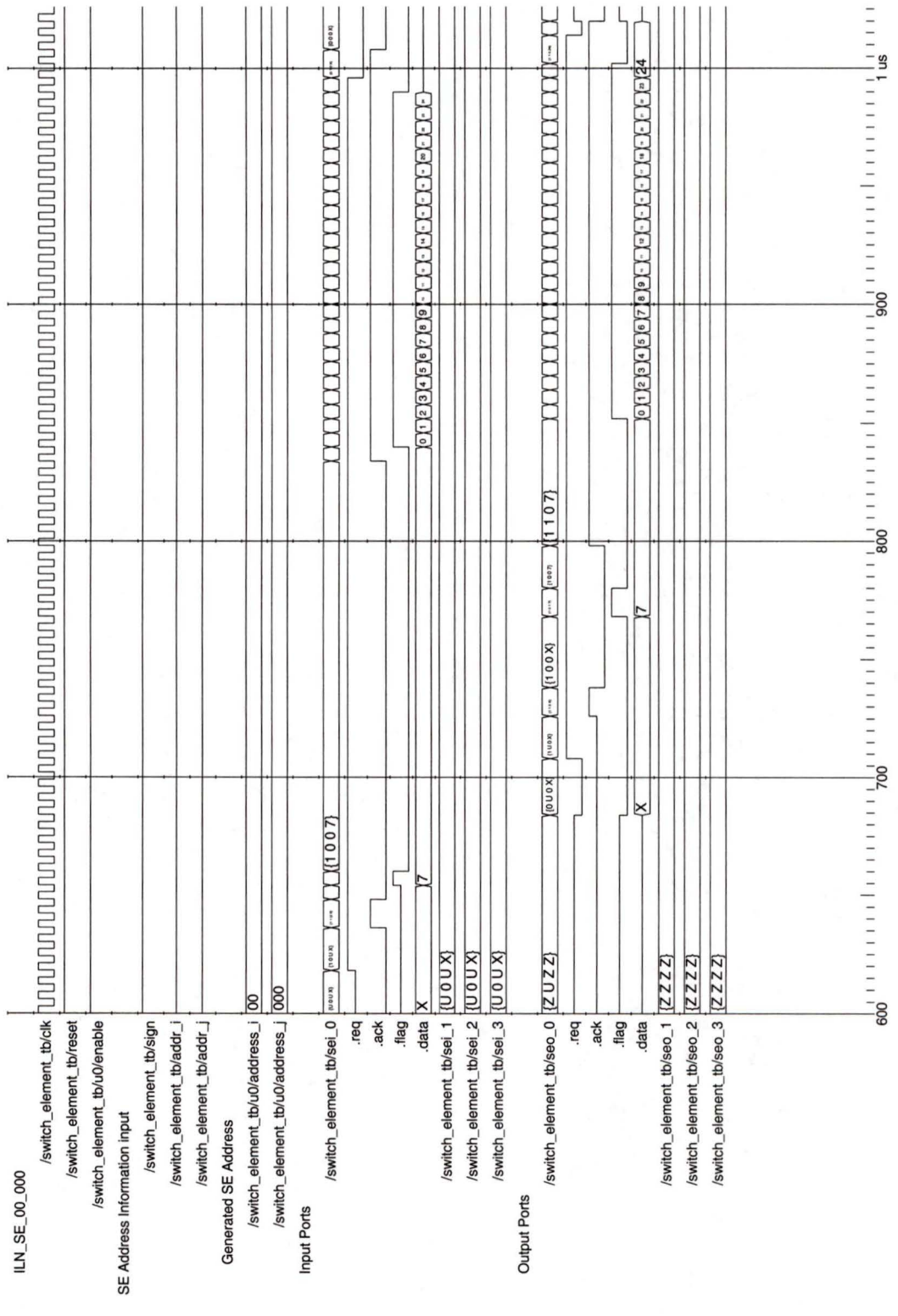


Figure 5.15. *ILN Network Packet transport process at SE(00,000)*

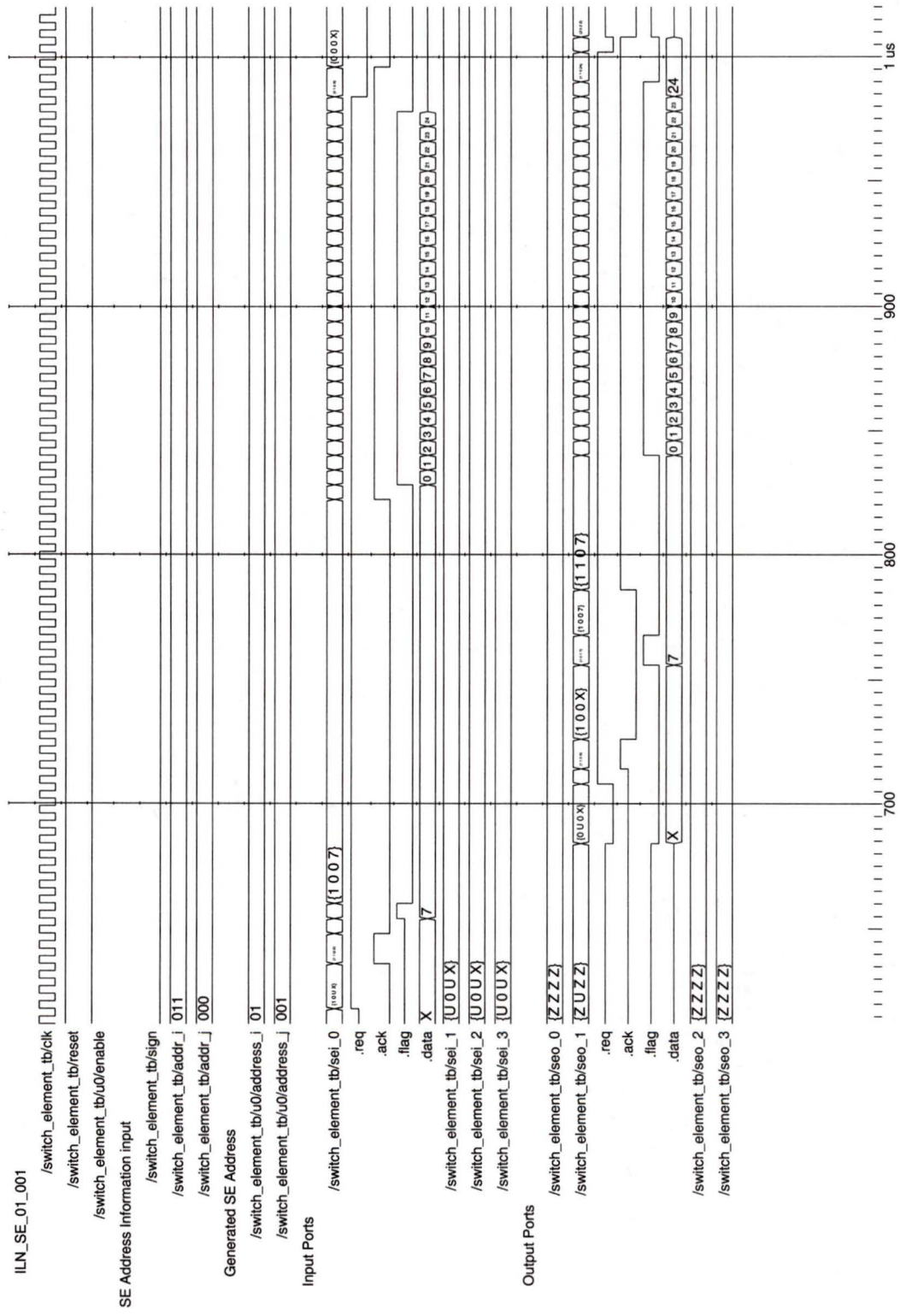


Figure 5.16. *ILN Network Packet transport process at SE(01,001)*

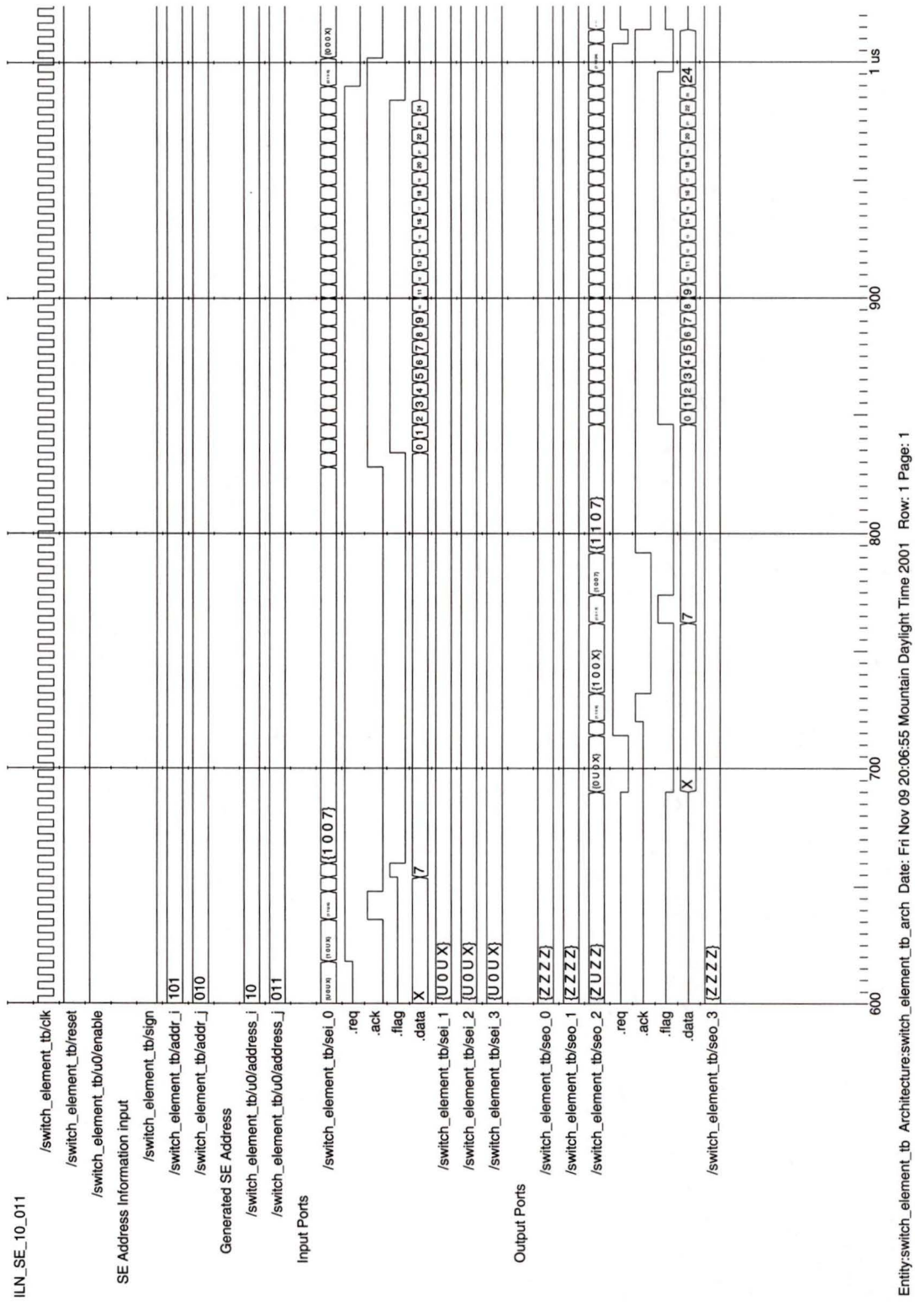


Figure 5.17. *ILN Network Packet transport process at SE(10,011)*

since it is not the ILN network's 0th SE. Refer figure 4.1, the SE-19 receives input addressing information from the SE-18; thus during the initialization phase, the SE receives preceding SE's addresses i & j as "101" & "010" over its input address lines. The bit-0 of input address i is '1', signifying the preceding SE is at the vertical location of the SE. Threfore, the SE's addressing function generates its addresses i & j as "10" and "011". During the packet switching phase, the SE when receives packet request and required destination address, it sends path establishment signals to the next SE over its output port SEO-2, in accordance to the created RV. Refer figure 4.1, within ILN network the SEO-2 of SE(10,011) is connected with the SE-31 (i, j: 11,111), being the next SE in this example.

The initialization and switching processes at ILN network's SE-31 (i, j: 11,111) are shown in Figure 5.18. In the figure, the SE Sign port is tied with '0' potential, since it is not the ILN network's 0th SE. Refer figure 4.1, the SE-31 receives input addressing information from SE-30; thus during the initialization phase, the SE receives preceding SE's addresses i & j as "111" & "110" over its input address lines. The bit-0 of input address i is '1', signifying the preceding SE is at the vertical location of the SE. Threfore, the SE's addressing function generates its addresses i & j as "11" and "111". During the packet switching phase, the SE when receives packet request and required destination address, it sends path establishment signals to the next SE over its output port SEO-3, in accordance to the created RV. Refer figure 4.1, within ILN network the SEO-3 of SE(11,111) is a straight connection with the ILN network port-7; which is the required destination. This ILN network's output port is further connected with the OM of the switch system.

5.6.2 ILN Network Synthesis Results

The Synthesis conventions and the target technology for ILN network hardware implementations are same as described in the SE synthesis section. All events of ILN network are synchronized to the rising edge of the global clock input signal.

A timing report of the synthesized logic for ILN network is shown in Table 5.2. The report show an estimated frequency of 112.3 MHz, giving 3.59 Gbps of data rate for 32-bit data lines. The worst case signal delay on a critical path is found as 8.9 ns. The average delay is calculated as 8.8 ns, and the set-up requirement as 0.1 ns.

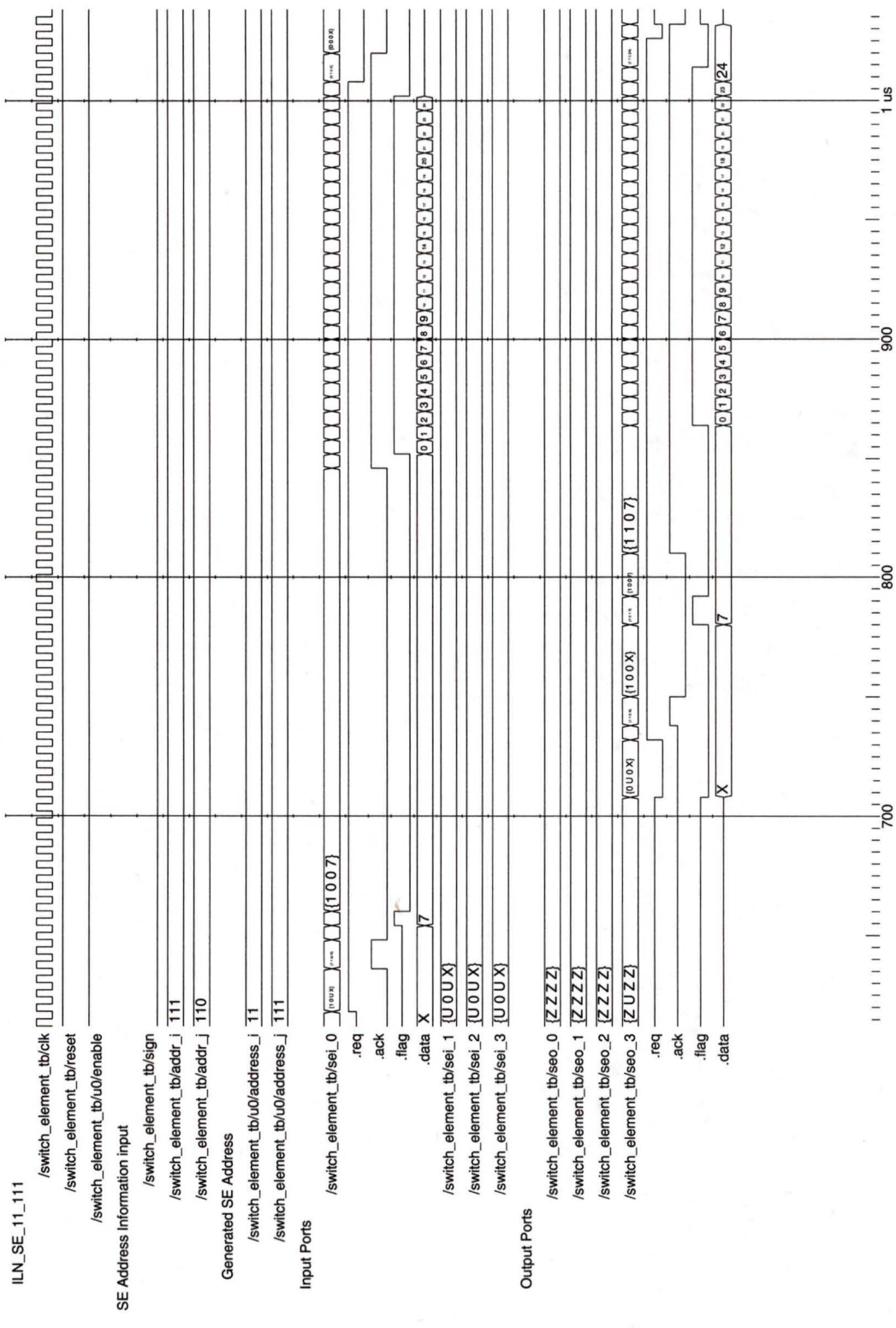


Figure 5.18. *ILN Network Packet transport process at SE(11,111)*

Table 5.2. *The ILN Network Timing Report*

Clock	Requested Frequency	Estimated Frequency	Requested Period	Estimated Period	Slack
Clk	144.0 MHz	112.3 MHz	6.9 Nano-Sec.	8.9 Nano-Sec.	-2.0

The timing report further shows a higher -2 value of slack, making the set frequency constraints and the estimated clock frequency off by 22%. The reason for this variance is due to non availability of adequate resources in the current technology device.

A summary of FPGA resource usage report for the synthesized ILN network logic design is reproduced below:

Cell usage:

BUF .... 5 used

FDE .... 8328 used

FD .... 1928 used

FDC .... 1433 used

FDP .... 32 used

GND .... 1 used

VCC .... 1 used

MUXF5 .... 3039 used

FDSE .... 32 used

FDR .... 5187 used

FDS .... 212 used

FDRS .... 32 used

XORCY .... 2912 used

MUXCY-L .... 2752 used

I/O primitives:

OBUF .... 280 used

IBUF .... 281 used

BUFGP .... 1 used

I/O Register bits .... 280 used

Register bits not including I/Os .... 16904 used (550%)

Internal tri-state buffer usage summary
BUFTs + BUFEs: 33 of 1536 (2%)
Global buffer usage summary
BUFGs + BUFGPs: 1 of 8 (12%)
Mapping Summary:
Total LUTs: 32602 (1061%)

The FPGA resource usage report suggests a requirement of 32602 LUTs for fully implementing the ILN network on the chip. This is equivalent to 2.8M system gates. The Virtex-II XC2V250 device has a total capacity of 3072 LUTs, being equivalent to 250K system gates. This result implies that the target device used in this synthesis has not enough capacity to accommodate the required system gates and register bits for the entire ILN network system. In this scenario, the design flow recommends to use a higher capacity device, and repeat the synthesis process until desired results are accomplished.

In this thesis work, due to the limitations in EDA tools licenses the higher capacity FPGA device could not be acquired. However, to study the method of synthesis the hardware implementation process for ILN network is completed in presence of known errors. In this process the VHDL design is optimized to its max, and is thus feasible for achieving desired results when applied on an appropriate capacity FPGA device.

5.7 Hardware Optimization Techniques

In this section specific characteristics of FPGA devices and their limitations are discussed in the light of experiences during the design work of ILN network and studies made in [59, 60, 64, 66, 70]. These mentioned techniques are like drop in the ocean of hardware design knowledge.

The following guidelines can provide a methodology to make optimum use of FPGAs in terms of design processing, speed and area:

- Use multiple levels of logic and common nodes as much as possible. Most synthesizers, if not all, cannot automatically generate nodes for common logic

sub-expressions. For example, if ten nodes are using the sub-expression A*B + C*D; then assigning it to a temporary node may not only save product terms, but may also save input terms to one or more FPGA blocks.

- Configuration cycles must work at the minimum hardware cost. Configuration cycles are usually performed only when the system boots, and are not important for high performance. Therefore, the logic for configuration should be optimized for space not speed.

- Pipeline the design. Pipelining is the process of splitting logic into stages so that the first stage can begin processing new inputs while the last stage is finishing the previous inputs. This ensures better throughput and faster circuit performance. For pipelining, the EDA tool splits the logic by moving registers of same FPGA row, same clock, and same control signal into the module.

- There are two methods to handle RAMs: instantiation and inference. The preferred method should be instantiating a RAM in the source code, rather than EDA tool infer from the behavioral code.

- When using both edges of the clock for registers, the logic for each logic should be coded in separate VHDL processes.

- In design optimization removal of unnecessary flip-flops dramatically increases the circuit frequency. In VHDL - the loop statement creates unnecessary flip-flops; a use of counters in design can optimize the creation of flip-flops,
- For synthesis in VHDL - wait clause within 'for' loop statement has no meaning in hardware and cannot be used to insert delay. A desired clock delay can be inserted by using flip-flops,
- In hardware design the circuit should be well defined in terms of logic components. To create a well defined circuit in VHDL, the loop statements should be made of fixed values rather than variable values,

- A flip-flop should always be at a known voltage on every clock edge to avoid metastable state. In VHDL the use of 'else' clause within 'if' statement will bring the created flip-flop to a known state.

5.8 Summary

This chapter presented the VLSI design work of ILN network, discussed hardware design techniques of industry, and then demonstrated the hardware implementation results for the ILN network. The design is implemented in Xilinx Virtex-II FPGA integrated circuit.

The chapter discussed a generic FPGA design flow along with the review of applied technology definitions. The signaling conventions used in the design are defined, demonstrated, and corresponding created logic is shown. Further, all the functions of ILN network are demonstrated and their relevant logic's timing and resource usage is illustrated.

It is shown that the main building block of ILN network, the Switching Element (SE) is successfully designed and implemented in the FPGA. The functions of SE are demonstrated as individually and within an ILN network. At design stage, due to the limitations on simulation EDA tool licensing, all the 32 SEs could not be knitted together to observe the SEs interaction within the ILN network. Similarly, due to the synthesis EDA tool licensing limitations a higher capacity FPGA device could not be used for PAR of the design. However, SEs timings within ILN network were verified in the synthesis tool, therefore there exist good confidence in the design functionality when all 32 SEs could be knitted in a higher capacity FPGA device.

The acquired synthesis results of SE and ILN network for 0.1 ns signal delay, clock frequencies of 156.2 MHz and 112.2 MHz, giving bandwidth capacity of 4.9 Gbps and 3.5 Gbps are well over the minimum limits for broadband communications [7, 4].

The chapter has further discussed the possible techniques to optimize a design. These observations are the results of solutions to cope the difficulties faced in this design work, and the research done in literature. These recommendations would act as pointers for a designer to remove unnecessary bottlenecks in the FPGA implementations of a design.

This chapter has proved with examples and statistics the implementation of ILN network theory in the real world. It has been demonstrated with hardware specifications and design implementation that ILN Network is capable of performing its functions correctly, and is a Fault Tolerant network.

Chapter 6

Conclusions and Future Work

6.1 Thesis Conclusion

An efficient broadband switch requires implementation of a high QoS and fault tolerant switch fabric within its design. The Improved Logical Neighborhood (ILN) network is a proven switch fabric which attains high throughput, low CLP and Delay, and high performance under faults. The research work of this thesis includes review of related concepts, theory, definitions, design and hardware implementation of the ILN switching fabric.

In the beginning of thesis a global view of a switch design is studied, highlighting its fundamental building blocks. These basic blocks included input and output packet buffers, switch fabric, and, control and management components. The study briefly describes the packet buffering strategies, and other modules of the switch. The main study is then focused over switch fabric component of the switch. The switch fabrics are broadly divided into Time Division and Space Division switching systems. The study reveals that space division switching systems can provide more efficient and scalable networks.

A class of space division switching technique is known as Multistage Interconnection Networks (MIN). The MIN has proved to be well suited in communication systems as it offers a good balance between cost and performance. MINs are composed of modular switching processors which are interconnected and interacts with each other under certain defined MIN classes and functions. These postulates create various MIN topologies offering different switching solutions. The study of this thesis then specifically discusses some of popular topologies - Generalized Cube Network, Banyan Network, and Data Manipulator networks, respectively.

The Improved Logical Neighborhood (ILN) network belongs to the family of MIN, and is based on the concepts of Data Manipulator Networks. The study discusses ILN network theory, and its constituent building elements. The switching process in ILN network for routing packets from an originating to the destination port involves path establishment and packet data transport phases. This is being achieved by applying built-in routing algorithms: Dynamic Routing Algorithm, and Bitwise Routing Algorithm. The study further reviews the ILN network performance analyses, being already done in the previous research works. In this review it has been proved that ILN switch fabric outperforms other networks.

The hardware design of ILN network is the focus of this thesis. The design employs IC industry's generic Design Methodology. Using this methodology, detailed specifications of all layers of the ILN network design are devised in this work. Using high level VHDL modeling and top-down structured VLSI Design Flow, these specifications are then applied to implement an ILN network hardware in the FPGA. The modular components used in the design interacts in timely manner with each other to perform all the functions of ILN network. The interaction of modules involve control signaling for path establishment and then transfer of actual packet data from input to output ports of ILN network.

The functional modules of ILN network are small and manageable components to provide for a simplified design process, and easy design changes. The switch fabric is simulated and synthesized to verify its logic functions and evaluate the resource allocation schemes. The results show confidence in the design while considering all the possible scenarios of switch processing. The resource utilization reports suggests the design is practically applicable for manufacturing in a real world. The Delay and clock speed attained by the design is well within limits for packet switching.

6.2 Thesis Contributions

In this thesis we presented a complete review of communication networks, with emphasis on the theory and research work for Improved Logical Neighborhood (ILN) network. The thesis further presents a hardware feasible digital logic design for the ILN switching fabric. The design is implemented in high level VHDL code which

covers all the technical details of the ILN switch fabric specifications. Finally, the thesis exhibits a VLSI implementation of the specified digital design. The work is carried out in the Xilinx Virtex-II FPGA while attaining desirable industry standard results for packet switching.

6.3 Suggested Future Work

The research work on ILN switching fabric can further be expanded for creating a marketable product by including the following options:

- enhance the fault tolerance complexities of the network by implementing BRA intelligent routing techniques.
- include the option of port scalability in the VHDL design of ILN network.
- complete VLSI re-implementation and re-optimization of ILN network using high capacity FPGA devices.
- Lab tests on the configured FPGA IC, using *ChipScope Integrated Logic Analyzer* EDA tool of Xilinx [71].
- performance analysis of the VHDL design of ILN network using OXD version of *Optimized Network Engineering Tools* (OPNET) [72].

Bibliography

[1] Andrew S. Tanenbaum, *Computer Networks*, Prentice-Hall, Inc., Upper Saddle River, NJ 07458, USA, 3rd edition, 1996.

[2] William Stallings, *Data and Computer Communications*, Prentice Hall, Upper Saddle River, NJ 07458, 5th edition, 1997.

[3] Mischa Schwartz, *Telecommunication Networks: Protocols, Modeling and Analysis*, Addison Wesley, 1987.

[4] CCITT, *Broadband Aspects of ISDN*, CCITT Recommendation I.121, Geneva, blue book fascicle iii.7 edition, 1989.

[5] IETF, *RFC 791: INTERNET PROTOCOL*, September 1981.

[6] Steven Lin and McKeown, "A simulation study of ip switching," *proc. ACM SIGCOMM conference*, pp. 15–23, 1997.

[7] McDysan David E., Spohn Darren L., *ATM: Theory and Application*, McGraw Hill, Inc., New York, NY 10020, 1994.

[8] Jean-Yves Le Boudec, "The asynchronous transfer mode: a tutorial," *Computer Networks and ISDN Systems*, pp. 279–309, 1992.

[9] Ra'ed Y. Awdeh, H. T. Mouftah, "Survey of atm switch architectures," *Computer Networks and ISDN systems*, pp. 1567–1611, 1995.

[10] Dr. Fayez El-Guibaly, *High Performnace Switches: Analysis and Design*, Northstar Digital Design, Inc., Victoria B.C. V9B 4X2, 1999.

[11] Timothy X. Brown, "A high-performance two-stage packet switch architecture," *IEEE Transactions on Communications, Vol. 47, No. 12*, pp. 1792–1795, December 1999.

[12] Rudiger H. Hofmann, and Rudi Muller, "A multifunctional high-speed switch element for atm applications," *IEEE Journal of solid-State Circuits, Vol. 27, No. 7*, pp. 1036–1040, July 1992.

[13] N. Ranganathan, Rajat Anand, and Girish Chiruvolu, "A vlsi atm switch architecture for vbr traffic," *Proc. of 11th Intl. Conf. on VLSI Design*, pp. 420–427, January 1998.

[14] Nader Mirfakhraei, "Design of a cmos buffered switch for a gigabit atm switching network," *IEEE Journal of Solid-State Circuits, Vol. 30, No. 1*, pp. 11–18, January 1995.

[15] Alain Chemarin, Alain Andre, Alain Botta, Jacques Majos, Jean-Luc Rainard, Henri Teyssier, and Pierre Thorel, "A high-speed cmos circuit for 1.2 gb/s 16 × 16 atm switching," *IEEE Journal of Solid-State Circuits, Vol. 27, No. 7*, pp. 1116–1120, July 1992.

[16] Kuochen Wang, and Ming-Howe Cheng, "Design and analysis of a growable multicast atm switch," *IEEE Transactions on Communications, Vol. 48, No. 7*, pp. 1091–1094, July 2000.

[17] A. Sriram and Fadi J. Kurdahi, "Behavioral modeling of an atm switch using speccharts," *Proc. of 9th Intl. Conf. on VLSI Design*, pp. 19–22, January 1996.

[18] B. Patel, F. Schaffa, M. Willebeek-LeMair, "The helix switch: a single chip cell switch design," *Computer Networks and ISDN Systems 28*, pp. 1791–1807, June 1996.

[19] Y.W.Deng, and W.T.Chen, "Efficient parallel multicast atm switch," *IEE Proc. Commun., Vol. 147, No. 2*, pp. 123–132, April 2000.

[20] Peter Newman, Greg Minshall, and Thomas L. Lyon, "Ip switching - atm under ip," *IEEE - ACM Transactions on Networking, vol. 6, no.2*, pp. 117–129, April 1998.

[21] Graeme B. Mund, *M.A.Sc. Thesis - Switching Element Design for BISDN*, University of Victoria, Victoria, B.C., Canada, 1992.

[22] Graeme B. Mund, Fayez El-Guibaly, "A 2 × 2 switching element for broadband isdn," *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 620–623, June 1989.

[23] James Andrew Gilderson, *M.A.Sc. Thesis - VHDL Design of an ATM Switch*, University of Victoria, Victoria, BC, Canada, 1995.

[24] J. Gilderson, F. El-Guibaly, V.K. Bhargava, "Vhdl design of an atm switch," *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 100–103, 1995.

[25] Mostafa Abd-El-Barr, Khalid Al-Tawil, and Osama Abed, "Fault tolerance and reliability analysis of multistage data manipulator networks," *International Conference*, pp. 275–280, 1995.

[26] A. Rayhan, A. Almulhem, S. Agarwal, F. El-Guibaly, "Fault tolerant atm switch using logical neighborhood network," *Informatica, vol. 23*, pp. 325–334, September 1999.

[27] Omar Hamdy Mohamed, *M.A.Sc. Thesis - ILN switching element design and performance measure*, University of Victoria, Victoria, BC, Canada, 1999.

[28] Carver Mead, Lynn Conway, *Introduction to VLSI Systems*, Addison-Wesley Publishing Co., Inc., Don Millls, ON, Canada, 1980.

[29] Neil H. E. Weste, Kamran Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley Publishing Co., Inc., Don Mills, ON, Canada, second edition, 1994.

[30] Michael G. Hluchyj, and Mark J. Karol, "Queueing in high-performance packet switching," *IEEE Journal on Selected Areas in Communications, Vol. 6, No. 9*, pp. 1587–1597, December 1988.

[31] Amr Sabaa, "Design and modelling of a nonblocking input-buffer atm switch," *Can. J. Elec. & Comp. Eng., Vol. 22, No. 3*, pp. 87–93, 1997.

[32] Mark J. Karol, Michael G. Hluchyj, and Samuel P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Transactions on Communications, Vol. COM-35, No. 12*, pp. 1347–1356, December 1987.

[33] Shigeki Shiokawa, Iwao Sasase, "Input and output queueing two stage atm switch with hot-spot route," *IEICE Trans. Commun., Vol. E81-B, No. 2*, pp. 194–198, February 1998.

[34] Joan Garcia-Haro and Andrzej Jajszczyk, "Atm shared-memory switching architectures," *IEEE Network*, pp. 18–26, August 1994.

[35] Noboru Endo, Takahiko Kozaki, Toshiya Ohuchi, Hiroshi Kuwahara, Shinobu Gohara, "Shared buffer memory switch for an atm exchange," *IEEE Transactions on Communications, Vol. 41, No. 1*, pp. 237–245, January 1993.

[36] Howard Jay Siegel, *Interconnection Networks for Large-Scale Parallel Processing*, Lexington Books - D.C. Heath and Company, Lexington, MA, U.S.A., 1985.

[37] Josá Duato, Sudhakar Yalamanchili, Lionel Ni, *Interconnection Networks: An Engineering Approach*, IEEE Computer Society, Los Alamitos, CA, U.S.A, 1997.

[38] Robert J. McMillen, "A survey of interconnection networks," *IEEE Global Telecommunications Conf.*, pp. 105–113, 1984.

[39] Howard Jay Siegel, Robert J. McMillen, and Philip T. Mueller, Jr., "A survey of interconnection methods for reconfigurable parallel processing systems," *Proc. AFIPS - National Computer Conference, Vol. 48*, pp. 529–542, 1979.

[40] Laxmi N. Bhuyan, Qing Yang, Dharma P. Agarwal, "Performance of multiprocessor interconnection networks," *IEEE*, pp. 25–37, February 1989.

[41] C. Clos, "A study of non-blocking switching networks," *Bell System Technology Journal, Vol. 32*, pp. 406–424, 1953.

[42] Mark A. Franklin, "Vlsi performance comparison of banyan and crossbar communications networks," *IEEE Transactions on Computers, Vol. C-30, No.4*, pp. 283–291, April 1981.

[43] George B. Adams III, Dharma P. Agarwal, Howard Jay Siegel, "A survey and

comparison of fault-tolerant multistage interconnection networks," *IEEE Computer, Vol. 20, No. 6*, pp. 14–27, June 1987.

[44] H.J. Siegel, S.D. Smith, "Study of multistage simd interconnection networks," *Fifth annual Int'l symp. Comp. Arch.*, pp. 223–229, April 1978.

[45] C.P. Kruskal and M. Snir, "The performance of multistage interconnection networks for multiprocessors," *IEEE Trans. Computers, vol. 32*, pp. 1091–1098, 1983.

[46] Clark D. Thompson, "Generalized connection networks for parallel processor intercommunication," *IEEE Transactions on Computers, Vol. c-27, No. 12*, pp. 1119–1125, December 1978.

[47] Anand R. Tripathi, G. Jack Lipovski, "packet switching in banyan networks," *Proc. 6th annual Symp. Computer Architecture*, pp. 160–167, June 1979.

[48] Komain Pibulyarojana, Shigetomo Kimuru, and Yoshihiko Ebihara, "A study on a hybrid dilated banyan network," *IEICE Trans. Commun., Vol. E80 B, No. 1*, pp. 116–126, January 1997.

[49] Janak H. Patel, "Processor-memory interconnections for multiprocessors," *IEEE Proc. 6th Annual Symposium Computer Architecture*, pp. 168–177, June 1979.

[50] C.W.Chen, N.P.Lu, T.F.Chen, and C.P.Chung, "Fault-tolerant gamma interconnection networks by chaining," *IEE Proc.-Comput. Digit. Tech., Vol. 147, No. 2*, pp. 75–81, March 2000.

[51] Yeonghwan Tscha and Kyoon-Ha Lee, "Yet another result on multi-$\log_2 n$ networks," *IEEE Transactions on Communications, Vol. 47, No. 9*, pp. 1425–1431, September 1999.

[52] Jin-Fu Li, Shyue-Kung Lu, Shih-Arn Hwang, and Cheng-Wen Wu, "Easily testable and fault-tolerant fft butterfly networks," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 47, No.9*, pp. 919–929, September 2000.

[53] L.R. Goke, and G.J. Lipvski, "Banyan networks for partitioning multiprocessor systems," *First Annual International Symposium on Computer Architeciure*, pp. 21–28, December 1973.

[54] T. Feng, "Data manipulating functions in parallel processors and their implementations," *IEEE Trans. Comp, Vol. C-23*, pp. 309–318, March 1974.

[55] Randy H. Katz, *Contemporary Logic Design*, The Benjamin/Cummings Publishing Co., Inc., Redwood City, CA 94065, 1994.

[56] Douglas Lewin, David Protheroe, *Design of Logic Systems*, Chapman & Hall, London, U.K., second edition, 1992.

[57] Franklin P. Prosser, David E. Winkel, *The art of Digital Design*, Prentice-Hall, Inc., Englewood Cliffs, NJ 07632, U.S.A., second edition, 1986.

[58] Model Technology Inc., *ModelSim User's Manual*, Mentor Graphics Corporation, 10450 SW Nimbus Avenue, Portland, OR 97223-4347, v5.5d edition, 2001.

[59] Synplicity, *Synplify Pro User Guide*, Synplicity, Inc., 935 Stewart Drive, Sunnyvale, CA 94085, U.S.A., February, 2001.

[60] Xilinx, *Xilinx Foundation Series - Integrated Synthesis Environment (ISE) User Guide*, Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124-3400, v3.1i edition, 2000.

[61] Xilinx, *Virtex-II Platform FPGA Handbook*, Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124-3400, U.S.A., v1.0 edition, December, 2000.

[62] S. Brown, R. Francis, J. Rose, Z. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publishers, 233 Spring St., New York, NY 10013-1522, May, 1992.

[63] Stephen Brown, Jonathan Rose, "Fppa and cplds architectures: A tutorial," *IEEE Design and Test of Computers, Vol. 13, No. 2*, pp. 42–57, Summer 1996.

[64] Douglas J. Smith, *HDL Chip Design*, Doone Publication, 7950 Hwy 72W, Madison, AL 35758, USA, 2000.

[65] Peter J. Ashenden, *The Designer's Guide to VHDL*, Morgan Kaufmann Publishers, Inc., 340 Pine Street, San Francisco, CA 94104-3205, 1995.

[66] J. Bhaskar, *A VHDL Synthesis Primer*, Star Galaxy Publishing, 1058 Treeline Drive, Allentown, PA 18103, second edition, 1998.

[67] F. ElGuibaly and Sandeep Agarwal, "Design and performance analysis of shift register-based atm switch," *Proc. IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing*, pp. 70–73, 1997.

[68] F. ElGuibaly, "Design and analysis of arbitration protocols," *IEEE transactions on Computers, Vol. 38, No. 2*, pp. 161–171, February 1989.

[69] Prithviraj Banerjee, *Parallel Algorithms for VLSI Computer-Aided Design*, Prentice-Hall, Inc., Englewood Cliffs, NJ 07632, U.S.A., 1994.

[70] E. Finkelstein and S. Weiss, "Implementation of pci-based systems using programmable logic," *IEE Proc. - Circuits Devices Syst., Vol. 147, No. 3*, pp. 171–174, June 2000.

[71] Xilinx, *ChipScope Integrated Logic Analyzer - user manual*, Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124-3400, chipscope software v4.2i edition, 2000.

[72] Irene Katzela, *Modeling and Simulating Communication Networks: a hands-on approach using OPNET*, Prentice Hall, Inc., Upper Saddle River, NJ 07458, USA, 1999.

VITA

Surname: Shaikh ***Given Names:*** Anjum

Place of Birth: Karachi, Pakistan ***Date of Birth:*** Mar. 03, 1960

Educational Institutions Attended

University of Victoria	1998 - 2002
N.E.D. University of Engineering & Technology, Karachi, Pakistan	1979 - 1983

Degrees Awarded

B.Eng.(Electronics)	N.E.D. University of Engineering & Technology	1984

PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my thesis to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis: VLSI DESIGN OF IMPROVED LOGICAL NEIGHBORHOOD NETWORK.

Auth[redacted]

ANJUM SHAIKH
August 9th 2002