THESIS ON INFORMATICS AND SYSTEMS ENGINEERING C48

DfT-based External Test and Diagnosis of Mesh-like Networks on Chips

VINEETH GOVIND

TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Engineering
Chair of Computer Engineering and Diagnostics

This dissertation was accepted for the defence of the degree of Doctor of Philosophy in Computer and Systems Engineering on October 13, 2009

Supervisor: Dr. Jaan Raik

Opponents: Prof. Einar J. Aas, Norwegian University of Science and Technology (NTNU), Trondheim, Norway
Dr. Johnny Öberg, Royal Institute of Technology (KTH), Stockholm, Sweden

Defence of the thesis: November 27, 2009

Declaration:
Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology has not been submitted before for any degree or examination.

/Vineeth Govind/

ISSN 1406-4731
ISBN 978-9985-59-939-6

INFORMAATIKA JA SÜSTEEMITEHNIKA C48

Testitavusel põhinev välise testi ja diagnoosi meetod kahemõõtmelistele kiipvõrkudele

VINEETH GOVIND

To my parents

Abstract

Future integrated systems will contain billions of transistors, composing tens to hundred of IP cores which are capable of delivering rich multimedia and networking services. As the number of devices in the integrated circuit increases, the limitations of traditional bus architectures are exposed. Networks on Chips (NoC) have emerged as the new backbone for chip integration. Due to their particular topology on one hand, and their scale on the other hand, it is crucial to develop a methodology that could handle testing of such new kind of architectures.

In this thesis we discuss an external test method which achieves high fault coverage. In addition to that the test method is scalable, and has a low overhead area. Secondly we propose the use of the functional test configurations with a goal to locate faults in individual links of the switches. Collapsing of link faults based on equivalent configurations was also proposed. Using this, a method was proposed that is capable of unambiguously diagnose a link fault in the network in a very short test application time. Experiments showed that, although working at higher abstraction levels, the method has a very high coverage for logic-level structural faults. The test configurations were modified to get a much shorter test time.

In addition, the work proposed a set of Design-for-Testability (DfT) techniques for application of test patterns from the external boundary of a Network-on-a-Chip (NoC). The work presented in this thesis has been published in a journal and presented in several international conferences.

Annotatsioon

Tuleviku elektroonikasüsteemid saavad koosnema miljarditest transistoritest ning sadadest tuumadest, mis võimaldavad rikkalike multimeedia ja võrguteenuseid. Samas süveneb integraalskeemide keerukuse tõusust tulenevalt traditsiooniliste siini-arhitektuuride piiratus. Kiipvõrgud on saamas uueks ühendusviisiks keerukate kiipide integreerimisel. Tulenevalt nende regulaarsest struktuurist, kuid samas suurest keerukusest, on äärmiselt oluline töötada välja spetsiaalne metodoloogia kiipvõrkude testimiseks.

Käesolevas dissertatsioonis esitame me valise testi metodoloogia, mis saavutab kõrge rikete katte. Meetod on skaleeruv ning ei nõua oluliselt täiendavaid riistvara-ressursse. Teiseks pakume me kasutada välja töötatud testikonfiguratsioone rikete lokaliseerimiseks kiibiühendustes. Dissertatsioonis pakutakse välja ka meetod vastavate rikete hulga kollapseerimiseks, kasutades rikete ekvivalentsusklasse. Nimetatud meetod on võimaline kiirelt ja üheselt lokaliseerima rikkeid kiipvõrgu ühendustes. Katsetulemused näitavad, et kuigi meetod töötab kõrgel abstraktsioonitasemel, saavutab ta väga kõrge struktuursete rikete katte. Töös pakutakse välja ka parendatud testikonfiguratsioonid testimise aja minimiseerimiseks.

Lisaks esitatakse testitava projekteerimise tehnikad testvektorite edastamiseks kiipvõrgu väliste servade kaudu. Käesolevas dissertatsioonis esitatud töö on avaldatud teadusajakirjas ning mitmetel konverentsidel.

Acknowledgments

First of all I would like to thank my supervisor Dr Jaan Raik for guiding me through this relatively new and exciting topic. He has helped me by advising me, especially at moments when I had run out of ideas regarding the thesis and for encouraging me to finish this thesis. He answered my endless queries and has helped me immensely all through the past 4 years of my doctoral studies.

I also express my sincere gratitude to Prof. Raimund Ubar for his valuable suggestions, support and for giving me the initial opportunity to come to Tallinn in 2003 to do my master thesis. I would also like to thank all my colleagues from Tallinn University of Technology especially from the department of Computer Engineering.
I am grateful to the director of Dept of Computer Engineering Dr. Margus Kruus for his support.

Moreover, I would like to acknowledge the organizations that have supported my PhD studies. Tallinn University of Technology, Archimedes foundation, Estonian IT foundation (EITSA), National graduate school for information and communication Engineering (IKTDK), Enterprise Estonia (Project ELIKO).

Finally I would like to express my gratitude to my parents who in spite of being in India have always supported and kept me motivated.

Vineeth Govind
Tallinn, October 2009

Table of Contents

List of Publications

Journals

1. Jaan Raik, Vineeth Govind, Raimund Ubar. Design-for-Testability-Based External Test and Diagnosis of Mesh-like NoCs. *IET Computers and Digital Techniques*, Vol. 3, Issue 5, pp. 476-486, September 2009.

Conferences

2. Jaan Raik, Raimund Ubar, Vineeth Govind. Test Configurations for Diagnosing Faulty Links in NoC Switches. *Formal Proceedings of the 12th IEEE European Test Symposium*, IEEE press, pp. 29-34, Freiburg, Germany, May 20-24, 2007.

3. Jaan Raik, Vineeth Govind, Raimund Ubar. An External Test Approach for Network-on-a-Chip Switches. *Proceedings of the IEEE Asian Test Symposium 2006*, Fukuoka, Japan, pp. 437-442, Nov. 2006.

4. Vineeth Govind, Jaan Raik, Raimund Ubar. A Generic Synthesizable NoC Switch with a Scalable Testbench. *Proceedings of the Baltic Electronics Conference,* Laulasmaa, Oct. 2006.

Workshops

5. Vineeth Govind, Jaan Raik. Design-for-Testability for Application of External Test Patterns in a NoC. *2nd Workshop on Diagnostic Services in Network-on-Chips - Test, Debug, and On-Line Monitoring, In conjunction with Design Automation Conference (DAC),* Anaheim, USA, June 9, 2008.

6. Jaan Raik, Vineeth Govind, Raimund Ubar. An External Diagnosis method for Network-on-a-Chip. *IEEE/ACM Design Automation and Test in Europe Workshop on Diagnostic Services in Networks-on-Chips – Test, Debug and On-Line Monitoring,* April 16-20, Nice, France.

7. Jaan Raik, Vineeth Govind, Raimund Ubar. RTL Test Point Insertion for Sequential Circuits. *Proc. of the International Workshop on Testability Assessment (IWoTA), Rennes*, France, Nov. 2, 2004.

PhD Forums

8. Vineeth Govind, Jaan Raik, Raimund Ubar. DFT for Application of External Test Patterns in a Network-on-a-Chip. *The 3rd Annual Conference of the Estonian Information and Communication Technology Doctoral School*, Voore, April 25-26, 2008.

9. Vineeth Govind, Jaan Raik, Raimund Ubar. Test Configurations for Diagnosing Faulty Links in NoC Switches. *The 2nd Annual Conference of the Estonian Information and Communication Technology Doctoral School*, Viinistu, May 11-12, 2007.

10. Vineeth Govind, Jaan Raik, Raimund Ubar. A Generic Synthesizable NoC Switch with a Scalable Testbench. *The 1st Annual Conference of the Estonian Information and Communication Technology Doctoral School*, Jäneda, May 12-13, 2006.

List of Abbreviations

ANOC	Asynchronous Network on Chip
ATPG	Automated Test Pattern Generation
BIST	Built-in Self/Test
DfT	Design-For-Testability
DMEM	Data memory
DUT	Device under Test
EATE	Embedded Automated Test Equipment
FAUST	Flexible Architecture of Unified System for Telecom
FF	Flip Flop
FPGA	Field Programmable Gate Array
FMAC	Floating- point /multiply-accumulator
FSM	Finite State Machine
GALS	Globally Asynchronous Locally Synchronous
HDL	Hardware Description Language
LAN	Local Area Network
IC	Integrated Circuits
IEEE	Institute of Electrical and Electronics Engineers
IMEM	Instruction memory
LFSR	Linear Feedback Shift Register
I/O	Input Output
IP	Intellectual Property
MUX	Multiplexor
MISR	Multiple Input Signature Register
NIU	Network Interface Unit
NoC	Networks on Chips
OCN	On Chip Network
OEM	Original Equipments Manufacturer
OFDM	Orthogonal Frequency-Division Multiplexing
QDI	Quasi Delay-Insensitive
QoS	Quality of Service
RIB	Router Interface Block
RTL	Register-Transfer Level
SAF	Store and Forward
SoC	System on Chip
SSI	Small-Scale Integration
UUT	Unit Under Test
VHDL	Very High Speed Integrated Circuit Hardware Description language
VLIW	Very Long Instruction Word
VLSI	Very Large Scale Integration
WAN	Wide Area Network

MSI	Medium-Scale Integration
BILBO	Built-In Logic Block Observer
CSTP	Circular Self-Test Path
FEF	Functionally Equivalent Faults

Chapter 1
Introduction

Microelectronics is the area of technology associated with and applied to the realization of electronic systems made of extremely small electronic parts or elements. Steady advances in technology have resulted in devices with hundreds of millions of transistors and subsequent scaling of chips. The reduction in size feature increases the probability that a manufacturing defect occurs in the chip. It takes only one faulty transistor to make the entire chip fail to function properly. Therefore it is necessary to test components at various stages during the manufacturing process. The cost of testing has become a major component of the design and manufacturing costs of new products.

1.1 Networks on Chips

According to the International Technology Roadmap for Semiconductors, we will soon be entering the era of a billion transistors on a single chip [1]. It is being stated that soon we will have a chip 50-100 nm comprising around 4 billion transistors operating at a frequency of 10 GHz [1]. Such a development means that in the near future we will probably be having devices with such complex functions from mere mobile phones to mobile devices controlling satellite functions. But developing such kind of chips is not an easy task as the number of transistors increases on-chip and so does the complexity of integrating them. System on Chip (SoCs) use shared or dedicated buses to interconnect the communicating on-chip resources. However these buses are not scalable beyond a certain limit. In this case the current interconnect infrastructure will become a bottleneck for the development of billion transistor chips. The design of such complex systems includes several challenges. One challenge is designing-on-chip interconnection networks that efficiently connect IP cores. Another challenge is application mapping that makes efficient use of available hardware resources. An architecture that is able to accommodate such a high number of cores, satisfying the need for communication and data transfers is the networks on chip architecture.

Around 1999 several research groups started to investigate systematic approaches to the design of communication part of SoCs [2]. It soon turned out that the problem has to be addressed at all levels from the physical to architectural to the operating system and application level. Hence, the term Network on Chip (NoC) is today used mostly in a very broad meaning, encompassing the hardware communication infrastructure, the middleware and operating system communication services and design methodology and tools to map applications onto a NoC.

The NoC–based Soc design utilizes two major concepts, different from those of the usual bus–based SoC architecture. First, it is a packet transaction rather than circuit transaction, and, secondly it is a distributed network structure, rather than a conventionally globally shared bus or a centralized matrix. Each of its functional

modules should be designed latency insensitive to support packet transactions. The benefits gained from packet transaction are improvements in operational speed, reliability of the interconnection links, and efficient utilization of links. Another advantage is that the electrical parameters of the existing NoC are not affected by addition of other NoC modules because of the structured characteristic of the NoC. The other benefits from a NoC based approach are the possibility to reuse the communication network. The switches, interconnects and the lower communication protocols can be designed, optimized, verified and implemented once and reused in a larger number of products. There is in fact a long list of services that would be benefiting many applications but can impossibly be developed from scratch for each new product. Examples are

- the detection, monitoring and management of faults in the network;
- the allocation and management of network resources and possibly task migration for load balancing and power optimization;
- the management of global and shred memory;
- the provision of sophisticated communication services such as channels with guaranteed bandwidth and quality of service, multi-cast and broadcast communication, etc;

The industry has initiated different NoC based designs such as AEthereal NoC from Philips, the STNOC from STMicroelectronics, and an 80 core NoC from Intel. The key design challenges of emerging NoC designs are the communication infrastructure, communication paradigm selection and application mapping optimization.
NoC is emerging as a revolutionary methodology to integrate numerous blocks in a single chip. It is the digital communications backbone that interconnects the components on a multicore SoC. It is well know that with shrinking geometries NoCs will be increasingly exposed to permanent and transient sources of errors that could degrade manufacturability, signal integrity and system reliability.

1.2 Test Issues in NoCs

Traditionally, correct fabrication of integrated circuits is verified by post manufacturing testing using different techniques ranging from scan-based techniques to delay and current-based tests [1]. Due to their particular nature, NoCs are exposed to a range of faults that can escape classic test procedures. Such faults include crosstalk, faults in the buffers of the NoC routers, and higher-level faults such as packet misrouting and data scrambling [3]. These faults add to the classic faults that must be tested post fabrication for all integrated circuits (stuck-at, opens, shorts, memory faults, etc.). Consequently, the test time of NoC based systems increases considerably due to these new faults.

Test time is an important component of the test cost and, implicitly, of the total fabrication cost of a chip. For large volume production, the total time that a chip requires for testing must be reduced as much as possible to keep the total cost low. The total test time of an IC is governed by the amount of test data that must be applied

and the amount of controllability/observability that the Design-for-Testability (DfT) techniques chosen by designers can provide. The test data increases with the complexity of the chip and size, so the option the DfT engineers are left with is to improve the controllability/ observability. Traditionally, this is achieved by increasing the number of test inputs/ outputs, but this has the same effect of increasing the total cost of an IC. DfT techniques, such as scan-based tests, improve the controllability and observability of IC internal components by serializing the test I/O data and feeding/extracting it to/from the IC through a reduced number of test pins. The trade-off is the increase in test time and test frequency, which makes at-speed test using scan-based techniques difficult. Although scan based solutions are useful, their limitations in the particular case of NoC systems demand the development of new test data generation and transportation mechanism that reduces the total test time and at the same time do not require an increased number of test I/0 pins.

An effective and efficient test procedure is, however, not sufficient to guarantee the correct operation of NoC data transport infrastructures during the lifetime of the IC. Defects may appear later in the life of an IC, due to causes like electro migration, thermal effects, material aging, etc. These effects will become more important with continuous dimension downscaling of the devices beyond 65nm and moving towards the nanoscale domain. The technology projections for the next generations of nanoelectronic devices show that defect rates will be in the order of one to ten percent and defect–tolerant techniques will have to be included in the early stages of design flow of digital systems [1]. Even with defect rates indicated by the International Technology Roadmap for Semiconductors (IRTS) for upcoming CMOS processes, it is clear that correct fabrication is becoming more and more difficult to guarantee. An issue of concern in the case of the communication-intensive platforms such as NoC is the integrity of the communication infrastructure.

The challenges of NoC testing lie in achieving sufficient fault coverage under a set of fault models relevant to NoC characteristics, under constraints such as test time, test power dissipation, low area overhead and test complexity. A fine balance must be achieved between test quality and test resources. To accomplish these goals, NoCs are augmented with design-for-test features that allow efficient test data transport, built–in test generation and comparison and post manufacturing yield tuning.

1.3 Motivation

Testing of NoCs is a new challenge to be overcome by the research community. Over the years, a number of NoC test approaches have been proposed. Vast majority of them are based on implementing design-for-testability structures (i.e. wrappers, scan-paths, built-in test pattern generators and compactors).

In [4], Aktouf proposes the use of boundary scan wrapper for NoC testing routers. However, Amory et al. [41] point out that the use of standard DfT solutions for networks-on-a-chip results in a prohibitively large area overhead. The authors of [5, 6] present a new scalable DfT method for NoC switches with only 8 % of overhead.

However, the test application times even for the smallest networks are measured in tens of thousands of clock cycles and the test pattern generation time does not scale: ten hours (!) is needed to test a 5x5 network. The obvious drawbacks of all of the above methods are the required silicon area for test structures and lack of support for at-speed and functional system testing. In [7], Petersen et al. introduce an idea of near-constant-time testable (C-testable) built-in self-test based test configuration. However, this idea has never been implemented, the required overhead area is prohibitively large and there is no reference of the test quality achieved.

What has been missing previously is a scalable external test approach relying mainly on the NoC network's own high-throughput infrastructure for test access. Test configurations to handle switch matrixes of reconfigurable hardware devices have been developed (e.g. [8]). However, despite apparent similarities, the ideas cannot be implemented for packet-switched routers. The main difficulty is that test configurations in NoCs are dependent on routing algorithms and thus it is not possible to activate arbitrary test paths similar to configuring FPGA switches.

1.4 Thesis Contribution

The main contributions of the thesis are outlined below

- A generic parametrizable VHDL description of a deflecting NoC switch was implemented and a benchmark family of 8 switches representing different possible a architecture configurations was synthesized

- A 3*3 network was synthesized. The data width of the Switches in the Network could be either 128 or 512 bit.

- An external functional test method for NoC switching networks is developed. The proposed algorithm allows to cover nearly all of the single stuck-at faults in the switching networks and also transition faults, opens and shorts at the interconnect lines.

- This method was modified to make it scalable with the size of the network and produce a test set whose volume grows linearly with the rank of the network.

- The impact of crossbar multiplexer implementation on fault coverage was analyzed. Some implementations of crossbar achieved fault coverage of 100% and the average fault coverage was 97.1%

- A method was developed that is capable of unambiguously diagnose a link fault in the network in a very short test application time.

- In order to implement the test algorithm in a NoC network, various DfT structures were devised. These include resource loopback for testing the

crossbar multiplexer of the resource connection, a modification to the control part to force YX routing, a compact logic BIST for the control unit and dedicated test logic for covering the enable signals of switch buffers.

1.5 Organization of the Thesis

The thesis is organized into six chapters:

- Chapter 1- Introduction
- Chapter 2- Fundamentals of Networks on Chip
- Chapter 3- Testing and Diagnosis of Digital Systems
- Chapter 4- Networks on Chips Testing Methods
- Chapter 5- External Test Approach for NoC Switches
- Chapter 6- Experimental Results and Conclusions.

Chapter 1 provides the necessary background for the concepts used in the paper. The chapter discusses the scaling of chips and evolution of Networks on chips briefly. Then a brief introduction to the necessity of NoC testing is given. This is followed by the motivation for the thesis.
Chapter 2 looks into the fundamentals of Networks-on-chips, outlines the benefits of NoC layered approach over bus technology and gives examples of industrial implementation of NoC.
Chapter 3 provides an overview of various aspects of VLSI testing.
Chapter 4 describes in detail various NoC testing approaches.
Chapter 5 explains the contribution of the thesis and explains the external test pattern application method, DfT structures, diagnosis.
Chapter 6 elaborates experimental results and the conclusions are summarized.

Chapter 2
Fundamentals of Networks-on-Chip

About this Chapter

Networks-on-chip is the latest development in VLSI integration. Increasing levels of integration resulted in systems with different types of applications, each having its own I/O traffic characteristics. Since the early days of VLSI, communication within the chip dominated the die area and dictated clock speed and power consumption. Using buses is becoming less desirable especially with the ever growing complexity of single-die multiprocessor systems. As a consequence, the main feature of NoC is the use of networking technology to establish data exchange within the chip. This chapter looks into the fundamentals of Networks-on-chips. The reason why the concept of networking is suitable for chips is examined first, followed, by trends in communication architectures.

The third subsection discusses the limitations of bus based architectures, followed by the advantages of using the NoC layered approach. The next two subsections deal with network architectures topologies, protocols respectively. The following subsections deal with queuing schemes, flow control and routing algorithms. The last two subsections describe an industrial and an academic NoC chip implementation.

2.1 Why On-Chip Networking?

SoCs require design methodologies that have commonalities with other types of large scale systems. The design methodologies in SoCs can be compared to the internet. The latter is capable of taming the system complexity and of providing reliable service in presence of local malfunctions .Thus the networking technology has been able to provide us with quality of service (QoS),despite the heterogeneity and the variability of the internet nodes and links. It is then obvious that networking technology can be instrumental for the bettering of VLSI circuit design technology.

On the other hand the challenges in merging the network and VLSI technologies are in leveraging the essential features of networking that are crucial to obtaining fast and reliable on-chip communication. However this does not mean using Network protocols like TCP/IP. This is not feasible due to the high latency related to the complexity of TCP/IP. On-chip communication should be fast so networking techniques must be simple and effective. Bandwidth, latency and energy consumption for communication must be traded off in search of the best solution.

VLSI chips have wide availability of wires on many layers which can be used to carry data and control information. Wide data busses realize the parallel transport of information. Moreover, data and control do not need to be transported by the same

means, as in networked computers. Local proximity of computational and storage unit on chip makes transport extremely fast. Overall, the wire-oriented nature of VLSI chips make on-chip networking both an opportunity and a challenge.

In summary, the main motivation for using on-chip networking is to achieve performance using a system perspective of communication. This reason is corroborated by the fact that simple on-chip communication solutions do not scale up when the number of processing and storage arrays on chip increases. For example, on-chip buses can serve a limited number of units and beyond that, performances degrades due to the bus parasitic capacitance and the complexity of arbitration. Indeed, it is the trend to larger scale on-chip multiprocessing that demands on-chip networking solutions.

2.2 Trends

Busses have successfully been implemented in virtually all complex System on Chip (SoC) Silicon designs. Busses have typically been handcrafted around either a specific set of features relevant to a narrow target market, or support for a specific processor.

Several trends have forced evolutions of systems architectures, in turn driving evolutions of required busses [9]. These trends are:

- Application convergence: The mixing of various traffic types in the same SoC design (Video, Communication, Computing and etc.). These traffic types, although very different in nature, for example from the Quality of Service point of view, must now share resources that were assumed to be “private” and handcrafted to the particular traffic in previous designs.

- Moore’s law is driving the integration of many IP Blocks in a single chip. This is an enabler to application convergence, but also allows entirely new approaches (parallel processing on a chip using many small processors) or simply allows SoCs to process more data streams (such as communication channels)

- Consequences of silicon process evolutions between generations: Gates cost relatively less than wires, both from an area and performance perspective, than a few years ago.

- Time-To-Market pressures are driving most designs to make heavy use of synthesizable RTL rather than manual layout, in turn restricting the choice of available implementation solutions to fit bus architecture into a design flow.

These trends have driven of the evolution of many new bus architectures. These include the introduction of split and retry techniques, removal of tri-state buffers and multi-phase-clocks, introduction of pipelining, and various attempts to define standard communication sockets.

However, history has shown that there are conflicting tradeoffs between compatibility requirements, driven by IP blocks reuse strategies, and the introduction of the necessary bus evolutions driven by technology changes : In many cases, introducing new features has required many changes in the bus implementation, but more importantly in the bus interfaces (for example, the evolution from AMBA ASB to AHB2.0, then AMBA AHB-Lite, then AMBA AXI), with major impacts on IP reusability and new IP design. Busses do not decouple the activities generally classified as transaction, transport and physical layer behaviors. This is the key reason they cannot adapt to changes in the system architecture or take advantage of the rapid advances in silicon process technology. Consequently, changes to bus physical implementation can have serious ripple effects upon the implementations of higher-level bus behaviors. Replacing tri-state techniques with multiplexers has had little effect upon the transaction levels. Conversely, the introduction of flexible pipelining to ease timing closure has massive effects on all bus architectures up through the transaction level. Similarly, system architecture changes may require new transaction types or transaction characteristics. Recently, such new transaction types as exclusive accesses have been introduced near simultaneously within OCP2.0 [10] and AMBA AXI [11] socket standards. Out-of-order response capability is another example. Unfortunately, such evolutions typically impact the intended bus architectures down to the physical layer, if only by addition of new wires or op-codes. Thus, the bus implementation must be redesigned. As a consequence, bus architectures can not closely follow process evolution, or system architecture evolution. The bus architects must always make compromises between the various driving forces, and resist change as much as possible.

In the data communications space, LANs & WANs have successfully dealt with similar problems by employing a layered architecture. By relying on the OSI model, upper and lower layer protocols have independently evolved in response to advancing transmission technology and transaction level services. The decoupling of communication layers using the OSI model has successfully driven commercial network architectures, and enabled networks to follow very closely both physical layer evolutions (from the Ethernet multi-master coaxial cable to twisted pairs, ADSL, fiber optics, wireless..) and transaction level evolutions (TCP, UDP, streaming voice/video data). This has produced incredible flexibility at the application level (web browsing, peer-to-peer, secure web commerce, instant messaging, etc.), while maintaining upward compatibility (old-style 10Mb/s or even 1Mb/s Ethernet devices are still commonly connected to LANs). Following the same trends, networks have started to replace busses in much smaller systems: PCI-Express is a network-on-a board, replacing the PCI board-level bus. Replacement of SoC busses by NoCs will follow the same path, when the economics prove that the NoC either:

- Reduces SoC manufacturing cost
- Increases SoC performance
- Reduces SoC time to market and/or NRE
- Reduces SoC time to volume
- Reduces SoC design risk

2.3 Limitations of Bus Based Architectures

Limitations of traditional bus based-architectures start to become apparent when numerous processing elements compete for communication resources [12]. A typical bus-based architecture will require the processing elements to wait, while other data transfers are being carried out. Waiting in turn increases energy consumption due to leakage currents and decreases performance. Increasing the speed of the bus does improve the performance, but in the chips of the future, with more increased number of processing elements switch technology is needed to provide a balanced system. Bus systems don't meet next-generation requirements for converged data, voice, or video networks.
The disadvantages of busses can be summarized as following

- It is not scalable. As nodes are added, performance degrades due to capacitance and parasitics. There is a practical limit to the number of components in a system using a bus as its communication scheme and this limit is not high.
- It is not very testable. In a bus there are many possible states for the interface between nodes and medium. A full test of the structure is almost impossible for a nigh number of nodes and high coverage.
- There could be important degradation of access time due to contention and arbitration. The low priority components may suffer severe delays. The whole bus must be designed for the worst case which may lead to inefficiencies.
- Another problem is the lack of modularity. This causes poor fault tolerance

Thus a new communication paradigm is a prerequisite in multi-processor SOC (MP-SoC) as well as VLSI circuits with a huge number of transistors.

2.4 NoC Layered Approach Benefits

A summary of the benefits of the NoC layered approach are listed below [9]:

- Separate optimizations of transaction and physical layers. The transaction layer is mostly influenced by application requirements, while the physical layer is mostly influenced by Silicon process characteristics. Thus the layered architecture enables independent optimization on both sides. A typical physical optimization used within NoC is the transport of various types of cells (header and data) over shared wires, thereby minimizing the number of wires and gates.

- Scalability. Since the switch fabric deals only with packet transport, it can handle an unlimited number of simultaneous outstanding transactions (e.g., requests awaiting responses). Conversely, Network Interface Unit [NIU] deals with transactions, their outstanding transaction capacity must fit the performance requirements of the IP Block or subsystems that they service.

However, this is a local performance adjustment in each NIU that has no influence on the setup and performance of the switch fabric.

- Aggregate throughput. Throughput can be increased on a particular path by choosing the appropriate physical transport, up to even allocating several physical links for a logical path.

- Quality of Service. Transport rules allow traffic with specific real-time requirements to be isolated from best-effort traffic. It also allows large data packets to be interrupted by higher priority packets transparently to the transaction layer.

- Timing convergence. Transaction and Transport layers have no notion of a clock: the clocking scheme is an implementation choice of the physical layer. NoC units are implemented in traditional synchronous design style (a unit being for example a switch or an NIU); sets of units can either share a common clock or have independent clocks. In the latter case, special links between clock domains provide clock resynchronization at the physical layer, without impacting transport or transaction layers. This approach enables the NoC to span a SoC containing many IP Blocks or groups of blocks with completely independent clock domains, reducing the timing convergence constraints during back-end physical design steps.

- Easier verification. Layering fits naturally into a divide-and-conquer design & verification strategy. For example, major portions of the verification effort need only concern itself with transport level rules since most switch fabric behavior may be verified independent of transaction states. Complex, state-rich verification problems are simplified to the verification of single NIUs ; the layered protocol ensures interoperability between the NIUs and transport units.

- Customizability. User-specific information can be easily added to packets and transported between NIUs. Custom-designed NoC units may make use of such information, for example “firewalls” can be designed that make use of predefined and/or user added information to shield specific targets from unauthorized transactions. In this case, and many others, such application-specific design would only interact with the transport level and not even require the custom module designer to understand the transaction level.

2.5 Network Architectures

The network architecture specifies the topology and physical organization on the interconnection network. The elements of network architecture are the processing and storage units, called the nodes, the switches and the physical links. Nodes are computational elements and storage arrays. Nodes can incorporate switches; often called routers. Physical links are wires which can be interrupted by repeaters that have the task to amplify the signal and to steepen its edges.
Network architectures can be classified into four groups according to their topology [13].

- **Shared-medium networks** have the least complex interconnect structure in which the transmission medium is shared by all the communicating devices. In such networks only one device is allowed to use the network at a time. An important issue here is the arbitration strategy that determines the mastership of the shared-medium network to resolve network access conflicts. Their limited bandwidth restricts their use in multiprocessors.
- **Direct networks** or point-to- point network is a popular network architecture that scales well to a large number of processors. It consists of a set of nodes, each one being directly connected to a subset of other nodes in the network. Each of these nodes may be different computational elements. A common component of these nodes is router, which handles message communication among nodes. Direct networks have been a popular interconnection architecture for constructing large-scale parallel computers.
- **Indirect networks** use switches to carry out communication between any two nodes. Each switch can have a set of ports. Each port consists of one input and one output link. A set of ports in each switch is either connected to processors or left open, whereas the remaining ports are connected to ports of other switches to provide connectivity between the processors. The interconnection of these switches defines various network topologies.
- **Hybrid networks** combine mechanism of the above three networks. Therefore, they increase bandwidth with respect to shared-medium networks and reduce the distance between nodes with respect to direct and indirect networks. However, for systems requiring very high performance, direct and indirect networks achieve better scalability than hybrid networks because point-to-point links are simpler and faster than shared-medium buses.

All modern day on-chip networks are buffered, i.e., the routers contain storage for buffering messages when they are unable to obtain an outgoing link.
An interconnection network can be defined by four parameters

- Topology
- Routing algorithm
- Message switching protocol
- Router micro-architecture

2.6 Networks on Chips

The Networks-on-Chips (NoC) architecture, as outlined in figure 1 provides the communication infrastructure for the resources .In this way it is possible to develop the hardware of resources independently as stand-alone blocks and create the NoC by connecting blocks as elements in the network Moreover, scalable and configurable network is a flexible platform that can be adapted to the needs of different workloads, while maintaining the generality of application development methods and practices.

A two dimensional mesh interconnection topology is the simplest from a layout perspective [2] and the local interconnection between resources and switches are independent of the size of the network. Moreover, routing in a two-dimensional mesh is easy resulting in potentially small switches, high bandwidth, short clock cycle, and overall scalability. A NoC consists of resources and switches that are directly connected such that resources are able to communicate with each other by sending messages. A resource is a computation or storage unit. Switches route and buffer messages between resources. Each switch is connected to four neighboring switches through input and output channels. A channel consists of two one–directional point-to-point buses between two switches or a resource and a switch. Switches may have internal queues to handle congestion.

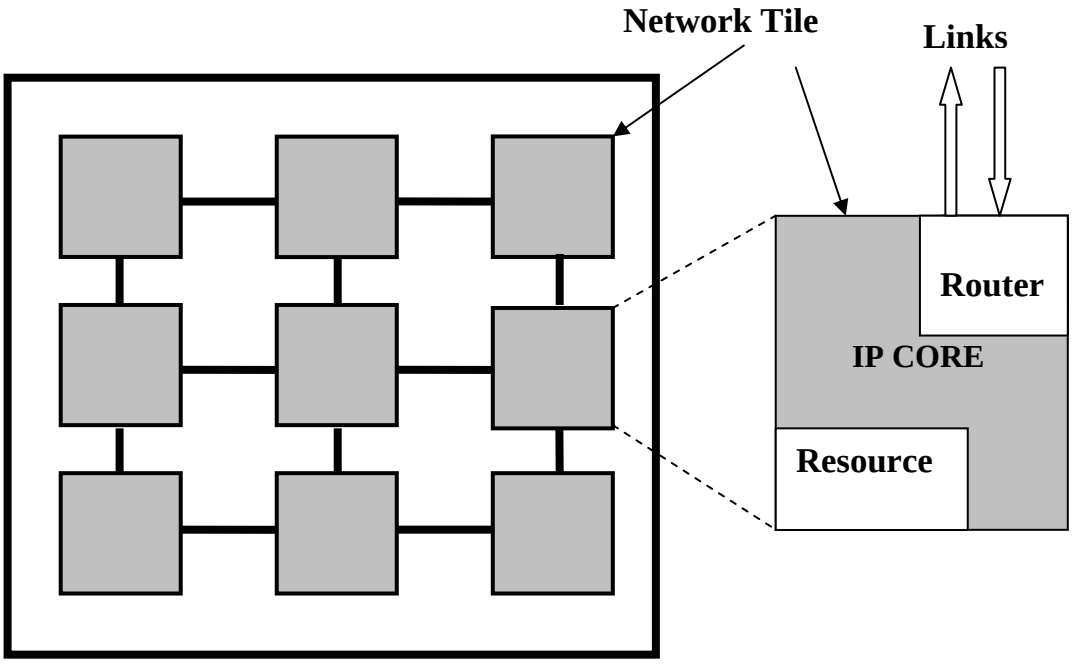


Figure 1: Network on chip architecture

Figure 2 illustrates an overall architecture of a NoC, and basic design issues are pointed out. First appropriate topology and protocol should be selected when NoC design begins. In the case of NoC topology, it can be configured with regular topologies such as Mesh, Torus or Star. Secondly protocols including packet format, end-to-end services, and flow control should be implemented in the network interface (NI) module. The packet switching scheme is the next factor to the determined. It is necessary that an appropriate switching scheme is chosen considering the target application and the silicon budget. Once the basic topology, protocol and routing method are determined, operation of the crossbar switch is as follows. When the input packets arrive at the input port, the crossbar switch scheduler gets the destination information from the input packets. If every packet arrives at a different input port and wants to leave from another output port, there are no input output conflicts. Then the scheduler connects the cross junctions so as to connect the packets at the input ports to their output ports. If conflicts occur, the scheduler should resolve them by predefined algorithm. Buffers are important to store packet data temporarily for congestion control.

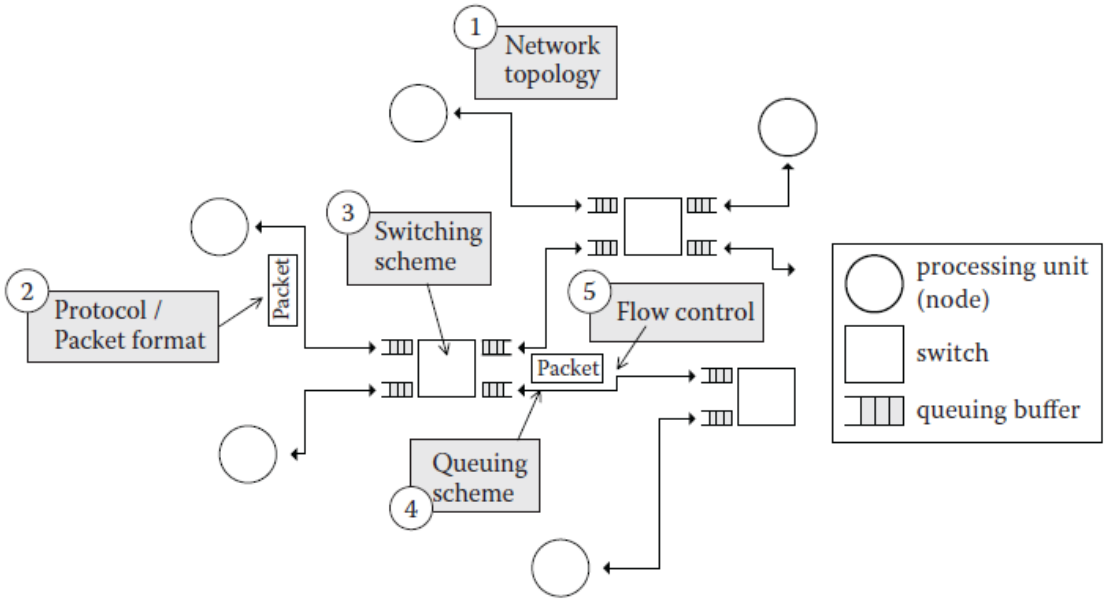


Figure 2: Basic design parameters of NoC

2.7 Network Topology

Network topology is the study of the arrangement or mapping of the elements (links, nodes). Topology is composed by an arbitrary number of instances of three basic kind of functional blocks [13].

- Network Interfaces
- Switches or Routers
- Links

Network Interfaces(NI) connect all the IP (Intellectual Property) cores to the network, mapping the bus type transitions coming from the IPs into packets that can be propagated inside the Network on chip and on the opposite side, building the bus transactions that correspond to packets that need to exit the NoC. The interface is just a wrapper that provides the computational unit a view of the network consistent with its I/O protocols. In other words the NI is a protocol converter that maps the processing node I/O protocol into the protocol used within the NoC. A reason for the need of this conversion is that NoCs use streamlined protocol for the sake of efficiency.

Switches (Routers) carry out the task of dispatching packets inside the network, depending on the particular routing scheme chosen. The number of ports depends on the topology of the network. Usually they have buffers or registers inside for storing information tokens, in order to minimize the risk of losing data due to congestion problems.

Links connect switches with network interfaces or with other switches. Links can be latency insensitive and can also contain buffering resources if needed by a particular application

The topology can be tailored and optimized for an application. More specifically, network topologies determine the number of hops and the wire length involved in each data transmission, both critically influencing the energy cost per transmission, performance and area. These factors are evaluated below.

- The silicon area of A NoC depends directly on the topology. The area of Network interfaces and routers sums up to the area cost of the links.
- The power dissipation of a NoC depends on the number of NoC components that are active, independently of data in the network and the power dissipation due to data in the NoC itself. Topologies with short routes and short links score well.
- Topologies with a high performance measure (e.g. small diameter) tend to have a high area cost (number of routers and links). Cost and performance must be traded off against one another.

NoCs differ from general networks because the router placement is limited to the two dimensional plane of the integrated circuit. Moreover links between the routers can travel only in X or Y direction.

Based on the network architecture the topologies can be divided as following

- Direct Networks
    - 1D: Linear, Ring
    - 2D: Mesh, Tree
    - 3D: Cube

- Indirect Networks
    - Crossbar,

    - Benes
    - Perfect shuffle
    - Omega
- Hybrid Network
    - Mesh star

Early NoCs used mesh and torus topologies for the sake of simplicity. A few of the common topologies are described below.

Ring topology is a network topology in which each node connects to exactly two other nodes, forming a single continuous pathway for signals through each node - a ring. Data travels from node to node, with each node along the way handling every packet.

Advantages of ring topology

- Very orderly network where every device has access to the token and the opportunity to transmit
- Performs better than a star topology under heavy network load.

Disadvantages of ring topology

- Since all the nodes are not directly connected, messages will have to hop along the intermediate nodes until they arrive at the final destination. This causes the network to saturate at a lower network throughput for most traffic patterns.

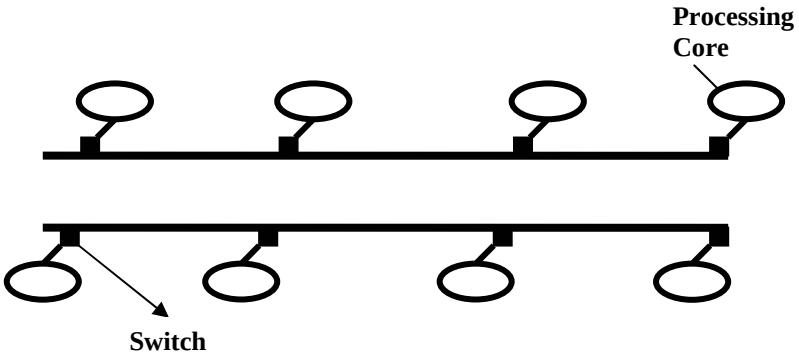


Figure 3: Ring topology

Star topology in its simplest form consists of one central router, which acts as a conduit to transmit data. Thus, the switch, nodes, and the transmission lines between them, form a graph with the topology of a star.

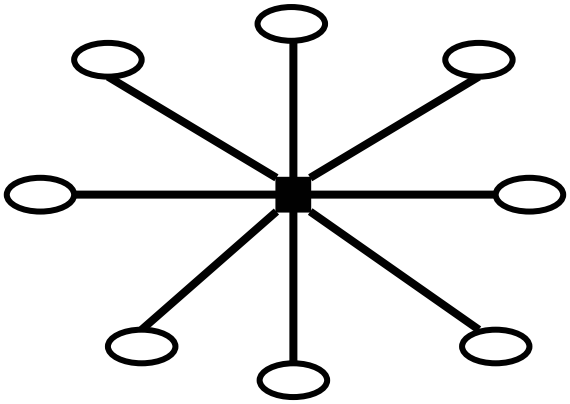


Figure 4: Star topology

Advantages of star topology

- **Better performance:** Passing of Data Packet through unnecessary nodes is prevented by this topology. At most 3 resources and 2 links are involved in any communication between any two resources which are part of this topology. This topology induces a huge overhead on the central router, however if the central router has adequate capacity, then very high network utilization by one device in the network does not affect the other resources in the network.
- **Isolation of devices:** Each device is inherently isolated by the link that connects it to the router. This makes the isolation of the individual devices fairly straightforward, and amounts to disconnecting the device from the router. This isolated nature also prevents any non-centralized failure from affecting the network.
- **Benefits from centralization:** As the central hub is the bottleneck, increasing capacity of the central router or adding additional devices to the star, can help scale the network very easily. The central nature also allows the inspection traffic through the network. This can help analyze all the traffic in the network and determine suspicious behavior.
- **Simplicity:** The topology is easy to understand, establish, and navigate. The simple topology obviates the need for complex routing or message passing protocols.

Disadvantages of Star topology

The primary disadvantage of a star topology is the high dependence of the system on the functioning of the central router. While the failure of an individual link only results in the isolation of a single node, the failure of the central router renders the network inoperable, immediately isolating all nodes. The performance and scalability of the network also depend on the capabilities of the router. Network size is limited by the number of connections that can be made to the router, and performance for the entire network is capped by its throughput.

2D Mesh Topology consists of switches in which each switch is connected to one resource and to four other switches.

Advantages of 2D mesh

- Although the resources may be of different size, the regularity of the mesh structure allows for fine control of physical parameters, predictability of performance, power and sophisticated, efficient clocking schemes.

Disadvantage of 2D mesh

- The area of mesh grows linearly with the number of nodes.

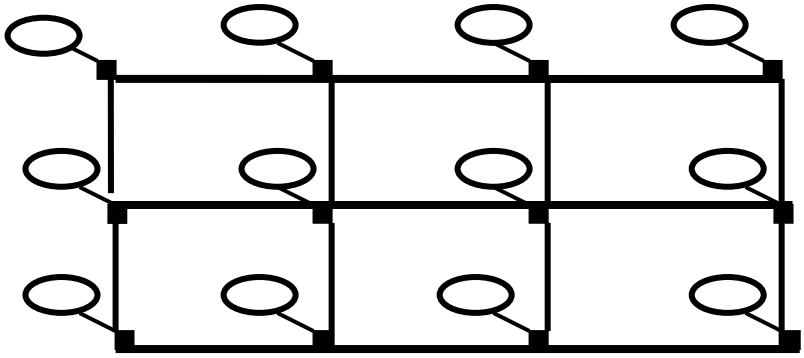


Figure 5: Mesh topology

Tree topology is a network topology in which a central 'root' node (the top level of the hierarchy) is connected to one or more other nodes that are one level lower in the hierarchy.

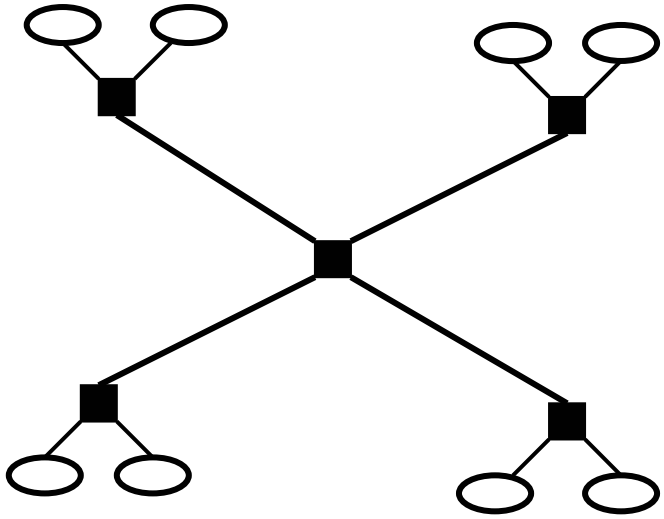


Figure 6: Tree topology

Advantages of tree topology

- A Tree topology is supported by many network vendors ad even hardware vendors.
- A point to point connection is possible with Tree Networks.
- All the computers have access to the larger and their immediate networks.
- Best topology for branched out networks.

Disadvantages of tree topology

- In a Network Topology the length of the network depends on the type of cable that is being used.
- The Tree Topology network is entirely dependant on the trunk which is the main backbone of the network. If that has to fail then the entire network would fail.
- Since the Tree Topology network is big it is difficult to configure and can get complicated after a certain point.

Other topologies include crossbar, hypercube, 2D torus, Spin, Octagon and hybrids of the above mentioned topologies.

2.8 Switching Schemes

Once the topology of a NoC has been decided on, the switching technique or how data flows through the routers must be determined. This involves defining the granularity of data transfer, and the switching technique.

Data is transferred on a link, which has a fixed width, measured in bits. The unit of data transferred in a single cycle on a link is called the phit (physical unit). Two routers synchronize each data transfer, to ensure that buffers do not overflow, for example. The unit of synchronization is called flit (flow control unit), and it s at least as large as a phit. Finally, multiple flits make up a packet, several of which may make up messages that modules connected to the NoC send to each other. There are two basic modes of transporting flits: circuit switching and packet switching.

Circuit vs. packet switching: In circuit switching a physical path from the source to the destination is reserved prior to transmission of the data. While in packet switching the data is partitioned and transmitted as fixed-length packets. The first few bytes of the packet contain routing and control information and are referred to as packet header. Each packet is individually routed from source to destination.

Packets are transferred to their destination through multiple routers along the routing path in a hop-by-hop manner. Each router keeps forwarding an incoming packet to the next router until the packet reaches its final destination.

While circuit switching reduces the network latency, it is done so at the expense of network throughput. Packet switching improves channel utilization and extends network throughput. There are three basic packet switching schemes [14].

1) Store-and-forward (SAF) Switching: Every packet is split into transfer units called flits .A single flit is sent from an output port of a router at each time unit. Once a router receives a header flit, the body flits of the packet arrive every time unit. To avoid input channel buffer overflow, the input buffer must be larger than the maximum packet size. The header flit is forwarded to the neighboring router after it receives the tail flit. The advantage of this technique is the simple needed control mechanism between routers due to the packet based operation. Its main drawback is the large needed channel buffer size that increases the hardware amount of the router. Moreover, SAF suffers from a large latency compared with other switching techniques, because a router in every hop must wait to receive the entire packet before forwarding the header flit. Thus, SAF switching does not fit well with the requirements of the NoCs.

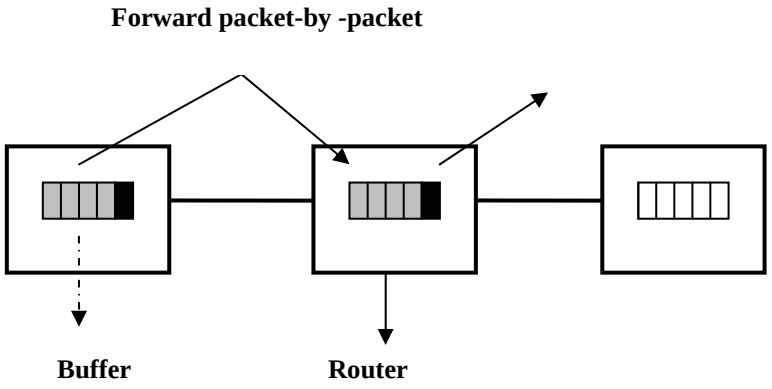


Figure 7: Store-and-forward switching

2) Wormhole (WH) Switching: Taking advantage of the short link length on a chip, an inter-router hardware control mechanism that stores only fractions of a single packet (flits).could be constructed with small buffers. Theoretically, the channel buffer at every router can be as small as a single flit. In wormholes (WH) switching, a header flit can routed and transferred to the next hop before the next flit arrives. Since each router can forward flits of a packet before receiving the entire packet, these flits are often stored in multiple routers along the routing path. WH switching reduces hop latency because the header flit processed before arrival of the next flits. Wormholes switching is better than SAF switching in terms of both buffer size and (unloaded) latency. The main drawback of WH switching is the performance degradation due to a chain of packet blocks. Since fractions of a packet can be stored across different routers along the routing path in WH switching, a single packet often keeps occupying buffers in multiple routers along the path, when the header of the packet cannot progress due to conflicts. Such a situation is referred to as a head-of –line (HOL) blocking. Buffers occupied by HOL blocking block other packets that want to go through the same lines, resulting in performance degradation.

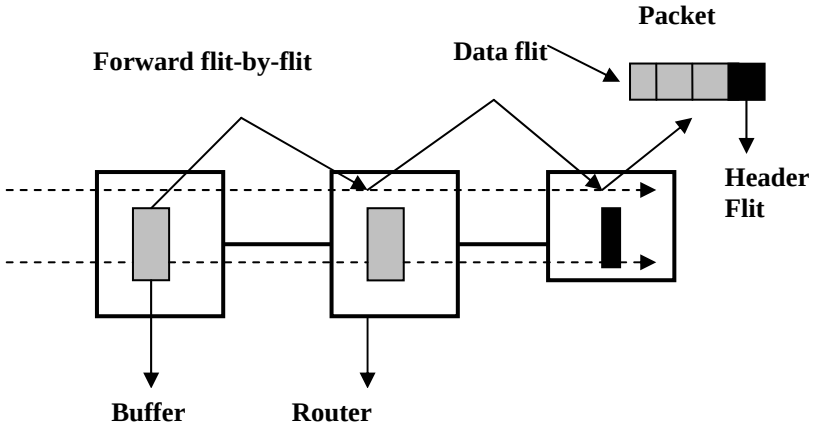


Figure 8: Wormhole switching

3) Virtual Cut-Through (VCT) Switching: To mitigate the HOL blocking that frequently occurs in WH switching, each router should be equipped with enough channel buffers to store a whole packet. This technique is called virtual cut-through (VCT), and can forward the header flit before the next flit of the packet arrives. VCT switching has the advantage of both low latency and less HOL blocking.

A variation called asynchronous wormhole (AWH) switching uses channel buffers smaller than the maximum used packet size (but larger than the packet header size). When a header is blocked by another packet at a router, the router stores the flits. Flits of the same packet could be stored at different routers. Another variation of VCT switching customized to NoC purposes is based on a cell structure using a fixed single flit packet.

2.8.1 Virtual Channels

The preceding switching techniques were described assuming that messages or parts of messages were buffered at the input and output of each physical channel. Buffers are commonly operated as FIFO queues. Therefore, once am message occupies a buffer for a channel, no other message can access the physical channel, even if the message is blocked, alternatively, physical channel may support several logic or vital channels multiplexed across the physical channel. Each unidirectional virtual channel is realized by independently managed pair of message buffers. Consider wormhole switching with a message in each virtual channel [14], each message can share the physical channel on a flit-by-flit basis. The physical channel protocol must be able to distinguish between the virtual channels using the physical channel. Logically each virtual channel operates as if each were using a distinct physical channel operating at the half the speed. Virtual channels can be used to improve message latency and network throughput.

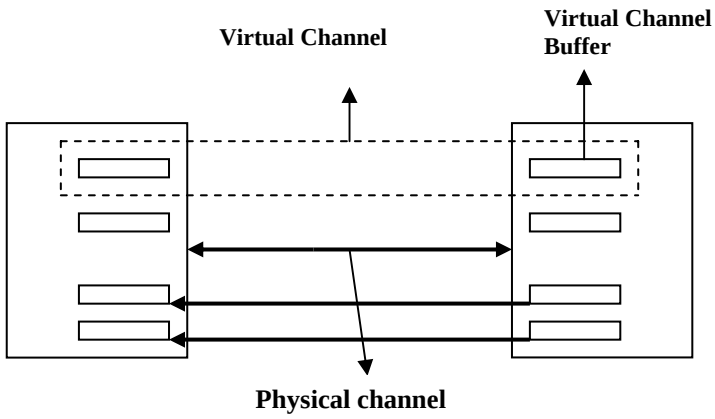


Figure 9: Virtual channels

2.9 Networks-on-Chip Protocols

A protocol is a set of conventions that governs the communication between two or more entities for interaction. The term entities stand for anything that is capable of transmitting or receiving information. For two or more entities to communicate with each other they must speak in a common language or, at least have appropriate translators; in addition, they must have proper rules to start, stop, and maintain communication.
The protocols used in the NoC strongly affect the performance, hardware amount and power consumption of on-chip interconnection networks [15]. High-quality communication that never loses data within the network is required for on-chip communication, because delayed packets of inter process communication may degrade the overall performance of the target application.
As the level of system of integration in SoCs began to rise, system designers faced the need to reuse to pre-designed and pre-verified computation units across different platforms and with different communication architectures. Therefore, the need for effective plug-and-play design styles pushed the development of standard interface sockets, allowing decoupling the development of computational units from that of communication architectures.
The open core protocol (OCP) is an openly licensed, core-centric protocol intended to ease contemporary system level integration challenges. The objectives of this standard are to enable processor core creation independently of network design, while sustaining parametrizeable design and high performance. Even though OCP is not meant to be a NoC protocol, it standardizes that design of Network Interfaces (NIs), as they do not need to be processing–core specific but just translators between OCP and the NoC protocol.

2.10 Queuing Schemes

There are three queuing schemes distinguished by location of the buffers inside the router [16].
1) Input queuing: Every incoming link has a single input queue so that N queues are necessary for N*N switches. Input queuing suffers from the head-of-line blocking problem; i.e., the switch utilization gets saturated at 58.6% load.
2) Output queuing: The queues are placed at the output port of the link, but N output queues for every outgoing link are required to resolve the output conflict, resulting in N^2 queues. Due to the excessive number of buffers and their complex wiring, in spite of its optimal performance, output queuing is not used.
3) Virtual output queuing: The advantages of the input queuing and output queuing are combined. A separate input queue is placed at each input port for each output requiring N^2 buffers. The head of-line blocking problem occurring in input queuing is resolved by scheduling. Complicated scheduling algorithms are required for this scheme.

2.11 Flow Control

There are several solutions for preventing packets from causing output conflict and buffer overflow [16].
1) Packet discarding: Once the buffer overflows, the packets coming again are simply dropped off.
2) Credit-based flow control: A back pressure scheme uses separate wires for the receiver to inform the transmitter of buffer congestion so as to prevent packet loss. The propagation delay time between transmitter and receiver should be considered carefully to avoid packet transmission while the wait signal is coming on the wire.
3) Rate-bases flow control: The sender gradually adjusts packet transmission according to the control flow messages from the receiver.

2.12 NoC Routing

Routing algorithms establish the path followed by each packet between source and the target switch. They must prevent deadlock, livelock and starvation situations [14]. Deadlock may be defined as a cyclic dependency among nodes requiring access to a set of resources, so that no forward progress can be made, no matter what sequence of events happens. Livelock refers to packets circulating the network without ever making any progress towards their destination. Starvation happens when a packet in a buffer requests an output channel, being blocked because the output channel is always allocated to another packet. Unlike traditional communication or interconnection networks, NoCs need not follow rigid networking standards.

2.12.1 Taxonomy of routing algorithms

Routing algorithms can be classified according to the three different criteria [14]:
(1) Where the routing decisions are taken: According to where routing decisions are taken, it is possible to classify the routing in *source and distributed routing*. In source routing, the whole path is decided at the source switch, while in distributed routing each switch receives a packet and defines the direction to send it. In source routing, the header of the packet has to carry all the routing information, increasing the packet size. In distributed routing, the path can be chosen as a function of the network instantaneous traffic conditions. Distributed routing can also take into account faulty paths, resulting in fault tolerant algorithms.
(2) How a path is defined: Depending how a path is defined, routing can be classified as *deterministic (oblivious) or adaptive*.

In *deterministic routing*, the path is completely specified from the relative position of source and target addresses. This routing scheme does not take into account the current load of the network links and routers when making routing decisions. It is simpler to implement in terms of router logic and interaction between routers and is more appropriate when traffic requirements are steady and known ahead of time.

In adaptive routing, the path is a function of the network instantaneous traffic. Adaptive routing increases the number of possible paths usable by a packet to arrive to its destination. This routing may use alternative paths when certain directions become congested and therefore have the potential of supporting more traffic using the same network topology.
(3) The path length: Regarding the path length criterion, routing can *be minimal or nonminimal*. Minimal routing algorithms guarantee shortest paths between source and target addresses. In nonminimal routing, the packet can follow any available path between sources and target Nonminimal routing offers great flexibility in terms of possible paths, but can lead to livelock situations and increase the latency to deliver the pack.

2.12.2 Virtual networks

A useful concept to design *routing algorithms* consists of splitting the network into several virtual networks [13]. A virtual network is a subset of channels that are used to route packets towards a particular set of destinations. The channel sets corresponding to the different virtual networks are disjoint. Depending on the destination, each packet is injected into a particular virtual network, where it is routed until it arrives at its destination. Virtual networks can be implemented using disjoint sets of virtual channels for each virtual network and mapping those channels over the same set of physical channels. Of course it is also possible to implement virtual networks by using separate sets of physical channels.

2.12.3 NoC dynamic routing schemes

Dynamic routing is an efficient alternative to balance the traffic load over a given NoC where the NoC traffic is unpredictable or changes with time [15]. The simplest method of dynamic routing is termed deflection routing or hot-potato routing. In this scheme, when a packet enters a router it will be sent toward a preferred output port according to a routing table or a routing function. However, if the preferred port is busy an alternative port will be selected. Here the router has no additional buffers in which to store the packets before they are moved, and each packet is constantly transferred until it reaches its final destination. The packet is bounced around like a hot potato sometimes moving further away from the destination because it has to keep moving through the network.

Deadlocks cannot happen in deflection routing when the number of input and output ports of a switch are identical and new local packets are not allowed in when all the inputs are busy. Livelocks may happen in deflection routing and simple priority rules can resolve it.

2.12.4 Deadlocks and livelocks of packet transfer

At the routing protocol layer of a computer network, packet may be dropped to allow forwarding another blocked packet. The figure 10 shows a situation where every packet is blocked by another one, and they cannot be permanently forwarded. Such a dependency is called a deadlock [14]. Once a deadlock occurs, at least a single packet within a network must be killed and resent. To avoid deadlocks, deadlock-free routing algorithms that can never cause deadlocks on path have widely been researched.

Besides the deadlock-free property a routing protocol must have the livelock-free property to stop packets from being discarded needlessly. Packets would not arrive at their destinations if they were to take nonminimal paths that go away from destination nodes. If this is the case, they would be permanently forwarded within NoCs. This situation is called livelock.

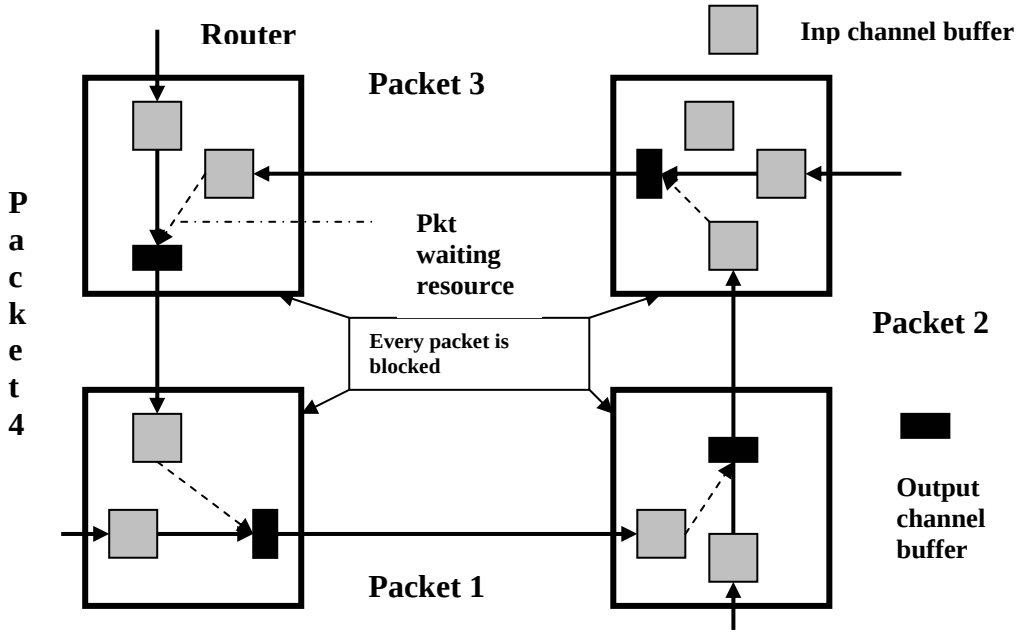


Figure 10: Deadlocks in routing protocols

The deadlock and livelock free properties are not strictly required in routing algorithms in cases of traditional LANs and WANs. This is because the Ethernet usually employs a spanning tree protocol that limits the topology to that of a tree whose structure does not cause deadlocks of paths; moreover the Internet Protocol allows packets to have a time-to-live field that limits the maximum number of file transfers. However, NoC routing protocols cannot simply borrow the techniques used by commodity LANs and WANs. Therefore new research fields dedicated to NoCs have developed, similar to those in parallel computers.

2.13 Real Chip Implementation of NoC

Silicon chip implementation trials for NoC based SoCs can be grouped into two categories: academic research and industrial approaches. Academic research shows complete chip implementations and demonstrations for specific applications. In the other hand, industrial approaches are mainly regarding the new protocol specifications, EDA tool chain and IP library support for the NoC developers. In the following subsections an industrial NoC and an academic implementation of NoC will be discussed respectively.

2.13.1 Tera-Flop 80-core NoC

Intel Corporation launched a terascale computing research program a few years ago to handle tomorrow's advanced applications, which would need a thousand times more computing capability than is available in today's giga-scale devices. For example there is real-time data mining across the teraflops of data, artificial intelligence for smarter cars and appliances; virtual reality for modeling, visualization, physics simulation and medical training.

The Tera-Flop NoC [17] consists of 80 processing cores that are connected through a 2D mesh packet switched on chip network. It performs up to 1 teraflop (10^{12} floating point operations) at 4GHz clock speed and consumes less than 100w.

Key enablers for Tera-Flop on a chip are listed below

- 80 Processing elements (PE), 160 single-precision floating point units (FPUs) designed for 4GHz operation
- Fast single-cycle accumulate loop
- Sustained FPU throughput: 2 FLOPS/cycle
- 80 Gbps router, operating at 4 GHz
- Shared and double pumped-crossbar switch
- 2-D mesh topology, 256 Gbps bisection bandwidth
- A 15 F04 (fan-out-of-4) balanced core and router pipeline
- Robust, scalable mesochronous clock distribution
- 65 nm eight-metal CMOS

2.13.1.1 NoC architecture overview

The NoC architecture contains 80 tiles arranged as 10*8 2-D mesh network and operating at 4 GHz. Each tile consists of a PE connected to a 5-port router with mesochronous interfaces, which forwards packets between tiles. The 80 tile On Chip Network (OCN) enables a bisection bandwidth of 256 Gbps. The PE contains two independent fully pipelined, single-precision, floating-point multiply-accumulator (FMAC) units with 3 KB single-cycle instruction memory (IMEM) and 2 BK data memory (DMEM). A 96-bit VLIW (Very long instruction word) encodes up to eight operations per cycle. With a 10 port (6-read, 4 –write) register file, the architecture allows scheduling to FMACs, simultaneous DMEM load and stores, packet send/receive from mesh network, program control, and dynamic sleep instructions. A router interface block (RIB) handles packet encapsulation between PE and router. The

fully symmetric architecture allows any PE to send (receive) instruction and data packets to (from) any other tile.

The 4 GHz 5-port wormhole-switched router uses two logical lanes: virtual channels for deadlock-free routing, and a fully non-blocking crossbar switch with a total bandwidth 80Gbps. Each lane has 16 FLIT queue, arbiter, and flow control logic. The router uses a five-stage pipeline with a two-stage round robin arbitration scheme that first binds an input port to an output port in each lane and then selects a pending FLIT from one of the two lanes.

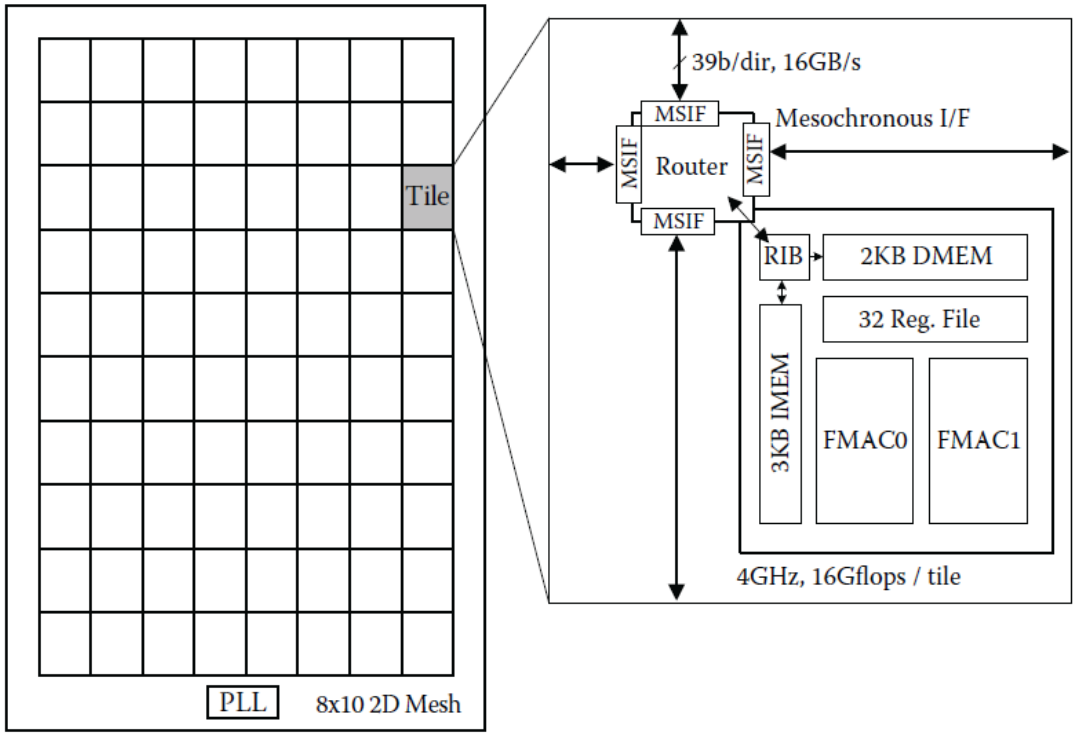


Figure 11: Intel Teraflop architecture

Each NoC packet is subdivided into multiple FLITS. Each packet has minimum two FLITs, and there is no maximum size limit. Each FLIT consists of a 6-bit control field and a 32-bit data field. The control field includes two flow control bits for each lane, a valid indication bit for the FLIT, and packet header/tail indication bits. There are three kinds of FLITs - header FLIT, PE control FLIT, and data FLIT. The header FLIT has 3 –bit destination ID (DID), which represents the out-port direction on each switching hop. Due to the data field size limit; the maximum hop count is limited to 10. However, a chained header seems to support larger hop counts. The PE control FLIT

includes address field and PE control information such as PE power management signals.

2.13.2 FAUST (Flexible Architecture of Unified System for Telecom)

Eleven European industrial research institutes and universities launched a joint project named "4-MORE-4G Multicarrier CDMA multiple antenna System ON Chip for Radio Enhancements". Recently, their architecture has been published for the application of multicarrier OFDM (Orthogonal Frequency-Division Multiplexing) based processing, such as 802.11n. They proposed the asynchronous Network on Chip (ANOC) with GALS (Globally Asynchronous Locally Synchronous) [18]. The ANOC architecture uses virtual channels to provide low latency and QoS (Quality of Service), which is implemented in quasi delay-insensitive (QDI) asynchronous logic.

The FAUST [19] chip integrates 20 asynchronous NoC routers, 23 synchronous units including ARM946 core, embedded memories, various IP blocks, reconfigurable datapath engines, and one clock management unit to generate 24 distinct unit clocks.

To integrate any synchronous IP within ANOC architecture, the dedicated NI performs two main tasks, it synchronizes the synchronous ands asynchronous logic domains using as hoc decoupling FIFOs, and provides all facilities to access the NoC communication infrastructure such as network routing path programming, network data packet generation, and IP core configuration.

The whole chip integrates more than 3 million gates and 3.5 Mb of embedded RAM. The maximum NoC throughput measured between two adjacent nodes or between and IP and its connected node are 5.12 Gbps per link.
Features of the FAUST Chip are listed below

- 2D-Mesh including 20 nodes
- Packet switching and wormhole routing
- Credit based flow control technique
- 32 bits word size
- 2 virtual Channels
- Asynchronous logic
- IP count -23 units connected to the NoC
- ARM94ES Processor core
- 24 on chip clocks
- NoC area is 15% of the global area

2.14 Summary

The challenge for chip designers is to come up with new architectures that achieve both a fast clock rate and high concurrency, despite slow wires. Shared bus networks are well understood and widely used in SoCs, but have serious scalability issues as more devices compete for the bus bandwidth. To mitigate this problem a new structured scalable on-chip communication fabrics, called networks–on-chip (NoC), has emerged for used in SoC designs. The basic concept is to replace the shared buses with on-chip packet-switched interconnection networks. NoC is emerging as a revolutionary methodology to integrate numerous blocks in a single chip.

In section 2.13 we considered two real –world examples of NoC implementations. While these examples of NoC implementations are more advanced than the switch benchmarks developed by us (see section 5.1) in terms of including queues, queue control and synchronizers. The test configurations presented in this thesis are still applicable to testing the crossbars of these NoC architectures.

Chapter 3
Test and Diagnosis of Digital Systems

About this Chapter

The introduction of integrated circuits (ICs), commonly referred to as microchip, was accompanied by the need to test these devices. Small-scale integration (SSI) devices, with tens of transistors in the early 1960s, and medium-scale integration (MSI) devices, with hundreds of transistors in the late 1960s were relatively simple to test .However, in the 1970s, large-scale integration (LSI) devices, with thousands and tens of thousands of transistors, created a number of challenges in testing these devices. In the early 1980s, very-large-scale integration (VLSI) devices with hundreds of thousand of transistors were introduced. Steady advances in VLSI technology have resulted in devices with hundreds of millions of transistors and many new testing challenges. This chapter provides an overview of various aspects of VLSI testing. Firstly the importance of testing, overview of faults and their classification is discussed. Then different types of fault model and the concept of fault coverage is discussed. This is followed by the distinguishing different testing methods. Then test generation method of combinational circuits, problems with test pattern generation for sequential circuits and taxonomy of test pattern generation for sequential respectively are discussed. The chapter ends with discussion of "Design for testability" and diagnosis.

3.1 Importance of Testing

Following the Moore's law, the scale of ICs has doubled every 18 months. The steady decreasing dimensions referred to as feature size, of the transistors and interconnecting wires from tens of microns to tens of nanometers, has made it possible to pack in more transistors in an IC, increasing operating frequency and clocks speeds. The reduction in feature size increases the probability that a manufacturing defect in the IC will result in a faulty chip [20]. A very small defect can easily result in a faulty transistor or interconnecting wire when the feature size is less than 100nm. Furthermore, it takes one faulty transistor or wire to make the entire chip fail to function properly or at the required frequency. Yet defects created during manufacturing process are unavoidable, and, as a result some number of ICs is expected to be faulty**;** therefore testing is required to guarantee fault free products.
Testing of a device can be defined as an experiment in which the device is exercised and its resulting response is analyzed to ascertain whether it behaved correctly. Testing is not only used to find fault-free devices but also to improve production yield at various stages of manufacturing by analyzing the cause of defects when faults are encountered. Testing is performed at various stages in the lifecycle of a VLSI device, including during the VLSI development process, the electronic system manufacturing process and, in some cases, system level operation.

3.2 Failures, Errors and Faults

A failure is a deviation in the performance of a circuit or system from its specified behavior and represents an irreversible state of a component such that it must be repaired in order for it to provide intended design function. A failure is caused by an error.

There is an error in the system when its state differs from the state in which it should be in order to deliver the specified service. An error is caused by a fault. A fault is present in the system when there is a physical difference between “good“ or “correct“ system and the current system

3.3 Fault manifestation

According to the way faults manifest themselves in time; two types of faults can be distinguished: permanent and non-permanent faults.
Permanent faults: refers to the presence of a fault that affects the functional behavior of a system (chip, array or board) permanently. Examples of permanent faults are:

- Incorrect connections between ICs
- Incorrect IC masks
- Functional design error

Non-permanent faults: refers to faults that are present only part of the time; they occur at random moments and affect the system functional behavior for finite, but unknown periods of time. Non-permanent faults can be divided into two groups with different origins: transient and intermittent faults.
Transient faults are caused by environmental conditions such as cosmic rays, pollution, humidity, temperature, vibration etc.
Intermittent faults are caused by non-environmental factors such as loose connections, deteriorating or ageing components, resistance and capacitance variations, physical irregularities, and noise.

3.4 Fault Models

Due to the diversity of VLSI defects, it is difficult to generate tests for real defects. Fault models are necessary for generating and evaluating a set of test vectors. Generally, a good fault model should satisfy two conditions (1) it should accurately reflect the behavior of defects. (2) It should be computationally efficient in terms of fault simulation and test pattern generation. However, the sheer number of defects that one may have to deal with under a fault model at this level may be overwhelming. To reduce the number of faults and hence the testing burden, one can go up the design hierarchy, and develop faults models which are perhaps less accurate, but more practical. No single fault model accurately reflects the behavior of all possible defects

that can occur. As a result, a combination of different fault models is often used in the generation and evaluation of test vectors and testing approaches developed for VLSI devices.

A good strategy is to first derive tests for the fault models at a higher level of abstraction, and then determine what percentage of faults at lower level are covered by the tests. Since a fault at higher level models many faults at a lower level, such a test set should cover a high percentage of faults at a lower level. However, due to imperfect modeling, many lower-level faults may remain undetected by this higher-level test set. Such faults can be specifically targeted at a lower level of design hierarchy.

Fault models have been developed at each level of abstraction, i.e. behavioral, functional, structural, switch-level and geometric.

3.4.1 Behavioral fault models

They are defined at the highest level of abstraction. They are based on the behavioral specification of the system. If the digital system described in a hardware description language, one could inject various types of faults in this description. The exact type of faults included in the behavioral model depends on the ease with which they allow detection of realistic faults at lower levels of abstraction. The importance of these models comes from the increasing desire among designers to start the synthesis process from a behavioral specification. Test sets derived from these fault models have been found to detect a high percentage (85 %)[21] of faults belonging to lower level of abstractions, such as-stuck at faults.

3.4.2 Functional fault models

Functional fault models are defined at the functional block level. They are usually ad hoc and geared towards making sure the functions of the functional block are executed correctly.
Consider a multiplexer .One can derive the following functional model for it.

- A 0 and 1 cannot be selected on each input line.
- When and input is being selected, another input gets selected instead of or in addition to the correct input

Another type of functional fault model assumes that the truth table of the functional block can change in an arbitrary way.

Efficient functional fault models have been developed for sequential circuit testing, memory testing and microprocessor testing [21].

3.4.3 Structural fault models

They assume that the structure of the circuit is known .Faults under these fault models affect the interconnections in this structure. The most well-known fault model under this category is the single stuck-at fault model. This is the most widely used fault model in the industry. The test patterns generated according to this fault models have an efficiency of up to 80-85 % [21]. However, the current trend is towards augmenting this fault models which allow detection of defects that stuck-at fault model is unable to cover.

3.4.4 Switch-level fault models

These kinds of fault models are defined at the transistor level. The most prominent fault models in this category are the stuck-open and stuck-on faults. If a transistor is permanently non-conducting due to a fault, it is considered to be stuck-open. Similarly, if transistor is permanently conducting, it is considered to be stuck-on. This fault model is specially suited for the CMOS technology.

3.4.5 Geometric fault models

These kinds of fault models assume that the layout of the chip is known. For example, knowledge of line widths, inter-line and inter-component distances and device geometrics are used to develop this fault model. At this level, one deals with opens and shorts. With shrinking geometries of VLSI chips, this fault model will become increasingly important.

3.5 Delay Fault Models

Instead of affecting the logical behavior of the circuit, a fault may affect its temporal behavior only; such faults are called delay faults. They adversely affect the propagation delays of signal in the circuit so the incorrect logic value may be latched at the output.
With the increasing emphasis on designing circuits for very high performance, delay faults are gaining wide acceptance.

3.6 Fault Coverage

The effectiveness of the test sets is usually measured by fault coverage. This is the percentage of detectable faults in the circuit under test (CUT) that are detected by the test set. The set is complete if its fault coverage is 100%.This level is desirable but rarely attainable in most practical circuits [22]. Moreover, 100% fault coverage does not guarantee that the circuit is fault-free [22]. The test checks only for failures than can be represented by the model used, such as a stuck-at/fault model. Other failures are not necessarily detected. The fault coverage is calculated using a fault simulator. This a logic simulator in which faults are injected at the appropriate nets of the

circuits, usually one at a time. The response of the circuits to test pattern applications is compared with good response of the circuit. The fault is considered detected if at least one the test patterns have a response different from the good circuit response.

3.7 Types of Testing

We can distinguish between various types of tests according to the test generation method.

3.7.1 Functional testing

Functional testing is used to verify that the model or logic behaves as it was intended. This is may also be called "Design validation test". If the design function represents an adder, then tests will be written to see whether the logic commits the necessary add functions and nothing more. Functional testing is measured by the logic comparing the known expected response to the applied stimulus. This kind of testing may also include timing or power consumption as part of the functional standard. In case of timing a second dimension is added to the behavior of the circuit: 1) that it conducts the correct behavior; 2) that it conducts this behavior to a timing standard. If a general purpose adder is placed within a chip design and it must be able to add two hexadecimal numbers with one clock cycle, the goal is to provide an additional set of tests that verifies that the slowest operation conducted by the adder is still faster than the target cycle time of the device. In case of power consumption, some defined operations must occur and the device can consume no more power than the specified power budget.

3.7.2 Structural testing

Structural testing is used to verify the topology of the manufactured chip. Given a good circuit before manufacturing process, structural testing can be used to verify that all connections are intact , and that all gate –level truth table are correct after the manufacturing process[21]. This type of testing can be done with reliance on static stuck-at fault model. Tests are developed by applying values to the inputs that toggle the suspected defective node to its opposite value and then applying values at the inputs that would allow the good value to propagate to a detect point. If the value at the detect point differs from the expected good value, then a fault has been detected. A delay fault model can be applied similarly to assess timing structure, and a current-based fault model can be used to assess power consumption.

3.7.3 Combinational exhaustive and pseudo-exhaustive testing

Combinational exhaustive testing is used to verify how the combinational portion of the model or logic behaves when every possible set of values is applied to the input ports, even if some of the vectors applied have no functional significance. Pseudo-exhaustive testing is the application of some portion of all possible logic values. Both these kinds of testing are done when a piece of logic is being characterized (timing or logic operation); when one type of model is being compared to another ;or when vectors will be applied in some manner without an understanding of the circuit. These tests are measured by the logic producing the correct logical or Boolean response to the applied stimulus and comparing the response to known ideal response or expected response.

3.7.4 Full exhaustive testing

It is the same as combinational exhaustive except that there are sequential elements that hold state embedded within the model or circuit. Merely applying every possible combinational value to the input pins is not enough to characterize a sequential design. There is the added complication of the applied sequence of all possible values of stimulus. A state machine with M elements requires 2^m tests to test all sequences. To fully test a combinational circuit with 2^n applied values, and to also consider all possible sequences, the combinational 2^n input values must be multiplied with 2^m state sequences resulting in $2^{(m+n)}$ tests. Again, this type of testing may be applied for characterization purpose, or when vectors are applied without an understanding of the circuit.

3.8 Test Pattern Generation for Combinational Circuits

Test pattern generation in the design process of generating appropriate inputs vectors to test a given digital design. Since exhaustive testing is usually prohibitively long unless steps are taken to partition the circuit or system into smaller parts, in which case exhaustive testing of each partition may become acceptable. However assuming that this cannot be or is not done, then some method of generating a reduced test (nonexhaustive) set has to be undertaken.
The generation of an acceptable reduced set of test vectors may be done in the following ways:

- Manual generation;
- Automated Test Pattern Generation

3.8.1 Manual test pattern generation is a method which the original circuit designer may adopt, knowing in detail the functionality of the circuit or system involved .The test patterns may be specified by considering a range of functional conditions, and listing the input test vectors and healthy output responses involved in

these situations. Alternatively the input vectors that will cause all the gates to switch at least once maybe considered.

This strategy of relying upon the circuit designer to propose some minimum set of test vectors can be a reasonable procedure to undertake for circuits containing say, a thousand but not tens of thousands of gates. It is a procedure which has been and still widely used in custom microelectronics (ASICs), where the IC vendor is manufacturing a specific circuit to meet the OEM's (Original Equipment Manufacturer) requirements. The vendor will take the OEM's suggested test vectors, and check that they are acceptable to both parties by performing some CAD simulation; this may take the form of checking how many of the internal nodes of the circuit are toggled by the suggested test vectors rather than considering the functionality of the tests. If this toggle coverage is considered inadequate (or incomplete), additional test vectors will be requested from the OEM to remedy the shortfall. It should be appreciated that the computer processing time for this procedure is relatively small, when compared to the excessive times which can build up when automatic test pattern generation is attempted.

3.8.2 Automated test pattern generation usually abbreviated to ATPG, becomes increasingly necessary as the gate count in the circuit increases to the thousands upwards. ATPG programs normally use a gate-level representation of the circuit, with all nodes or paths enumerated. In the semiconductor design and test support marketplace, many different tools are in use for vector generation. There are differences in the tools in their abilities and levels of support. Some tools are combinational-only and some tools are fully or partially sequential. ATPG tools also differ in the type of faults they support. The more common fault models are the stuck-at, transition delay, and path delay fault models for logic analysis , and pseudo-stuck – at and toggle for leakage analysis.

A complete and comprehensive automated vector generation process is made of several sub-processes, not just the ATPG tool. A good ATPG tool, however, may include several of these sub-processes as related features. The sub-processes can be broken down into two basic categories:

- Pre-ATPG
- ATPG

Pre-ATPG section includes the tasks of creating ATPG tool's library of standard cells, conditioning the design description so that the ATPG tool can operate on the provided format, and establishing the goals and constraints of the test process.

As an example, if an ATPG tool will operate on the EDIF (or Verilog or VHDL, or Proprietary format) description of a gate-level netlist, then a library must be made that describes the gates used in the EDIF format (e.g., flip-flops, AND-gates, OR-GATES etc.) in terms of the ATPG tool primitives. If constructs in the EDIF netlist can't be modeled or supported with the ATPG tool library, then the EDIF netlist itself must be modified (for example , areas comprised of raw transistors, or analog logic must somehow be represented by gate-level devices). All representations, in the library or

in the design description, must have identical Boolean behavior or else the created vectors may mismatch when simulated against other design formats [22].

Finally, before an ATPG tool can be used, the test process must be mapped onto the ATPG tool so that the vectors created will match the operation of the test platform. This step requires understanding the specific restrictions placed on the design by the tester, and by the design's own architecture. In most cases, this is a sequence file that describes the sequence and pin values required to place in test mode, and describes which signals or pin must have certain logic values applied during the sequence. This type of file is generally referred to as a "procedure" or a "constraint" file. Also required is a file containing the description of the legal observe points so that the test coverage metric is based on the observation made by the tester and not by " virtual test points" in a simulator.

ATPG process is the actual operation of the tool against the design description to generate vectors. This process may occur several times during the design process as a prototyping step to determine budget compliance (e.g., number of vectors, fault coverage), or the process may be applied several times against the final design description to generate the various individual vector sets needed to provide all the pattern sets for various test modes and constraints. All ATPG programs based upon fault models assume that a single fault is present when determining the test vectors. The usual fault model is stuck at fault model, which in practice covers a considerable number of other types of faults, but not all. The results of an ATPG program cannot, therefore, guarantee a defect-free circuit.

3.8.2.1 Algorithmic test pattern generation

A basic requirement in test pattern generation is to propagate a fault at a given node in the circuit to an observable output, such that the output is the opposite value in the presence of the fault compared with the fault-free output under the same input vector. This procedure may be termed path sensitising or forward driving. A second requirement is that the test input vector shall establish a logic value on the node in question which is opposite to the stuck-at condition under consideration.

Most test pattern generation algorithms but not all have as their underlying basis the following procedures, namely:

- Choose a faulty node in the circuit;
- Propagate the signal on this node to an observable output;
- Backward trace to the primary inputs in order to determine the logic signals on the primary inputs which correctly propagate this fault signal to the observable output

Here we consider two methods which have been used for test pattern generation, the first of which does not in fact use the above signal propagation procedure, and the second which uses this procedure.

I. **Boolean difference method:** is one test pattern generation technique which does not rely upon path sensitisation, but instead is functionally based using

Boolean algebraic relationships to determine the test vectors. The method is based upon the principle of using two Boolean expressions for the combinational network, one of which represents the fault-free behavior of the circuit, and the other the behavior under a single complementary fault condition. If these two functions are then exclusive-ORed together and the result is not a logic zero this fault can be detected; if the result is a logic zero, which means that the two functions are identical under this fault condition, then this fault cannot be detected.

II. **Roth's D-algorithm:** In contrast to the educationally-interesting Boolean difference method, Roth's D-algorithm [23] forms the underlying concept of many practical ATPG programs. It sensitises all paths from the site of a chosen fault to an observable output, and therefore inherently caters for reconvergent fan-out situations. It does, however, operate at the individual gate level, and requires knowledge of all the gates and their interconnection topology which functionally-based ATPG programs do not necessarily need.

   The D-algorithm involves five logic states, namely

   0= normal logic zero

   1= normal logic one

   D= a fault–sensitive state of a line or node, where D= 1 under fault-free conditions but is 0 under the particular fault condition being considered.

   Ď = a fault –sensitive state of a line or node, where Ď = 0 under fault-free conditions but is 1 under the particular fault condition being considered.

   X = an unassigned logic value, which can take any value 0, 1, D or Ď.

   Using these five logic states, the primitive D-cubes of failure for any type of logic gate may be defined.

III. **Other Algorithms:** Following the D-algorithm other methods have been developed. A few are listed here. PODEM [24] (the branch and bound search algorithm) which a heuristic successor of D-algorithm and FAN [25] (fan-out-oriented algorithm), many other techniques and heuristics have been developed, improving and speeding up the existing TPG systems. They are e.g. SOCRATES (using static and dynamic learning procedures) [26].

3.8.2.3 Pseudorandom test pattern generation

The ATPG algorithms previously discussed are deterministic, being based upon the choice and detection of a single stuck-at fault by an appropriate input test vector. Although each test considers one fault as the starting point for the determination of a test vector, each test vector covers more than one stuck-at node, and a final consolidation to the minimum number of test vectors to cover the complete stuck-at fault list can be implemented.

Deterministic ATPG algorithms provide the smallest possible test set to cover the given fault list. The disadvantage is the complexity and cost of generating this minimum test set. On the other hand a fully exhaustive test set will incur no ATPG costs, but will usually be too long to employ for large circuits. There is, however, an intermediate possibility which has been used. It is intuitively obvious that fault coverage increases with the number of input test patterns which are applied, up to the

full fault coverage; a single randomly chosen input test vector is also likely to be a test for several faults in a complex circuit .Hence if a sequence of random or pseudorandom input test vectors is used, it is probable that a number of circuit faults will be covered without incurring any ATPG cost. A truly random sequence of test vectors, which includes the possibility of the same vector occurring twice within a short sequence, is difficult to generate. Hence it is more practical to consider the use of pseudorandom patterns, which are very easy to generate. For a n-bit vector there can be a maximum of 2^n -1 sequences before the sequence repeats with each bit in the sequence having the same number of changes between 0 and 1.Hence 0s and 1s are equally probable on each bit.

Therefore, for a circuit with n primary inputs, it is appropriate to take a very small subset of the 2^n -1 pseudorandom sequence to use as the random test set. The number of faults that are covered by this test is determined by normal simulation, leaving the small percentage of faults which have not been detected to be covered by using some deterministic ATPG procedure.

3.9 Test Pattern Generation for Sequential Circuits

Combinational circuits can be flattened to gate level with stuck-at conditions considered at every node. In general, the latch and flip-flop elements of a sequential network cannot be broken down to individual gate level without breaking all the inherent feedback connections - if they could then the test pattern generation problem would become purely combinational with perhaps limited controllability and observability - and therefore we have to consider the possible states of the circuit as well as the combinational aspects [27]. The classic model for any sequential network is shown in figure 12. If the present primary outputs z_1 to z_m are a function of the present secondary inputs y_1 to y_s (the memory states) and present primary inputs x_1 to x_n, then the model is a Mealy model, if the outputs are a function of y_1 to y_s then the model is a Moore model. In both cases it is not possible to define the fault-free outputs resulting form a primary input test vector without knowledge of the internal states of the memory.

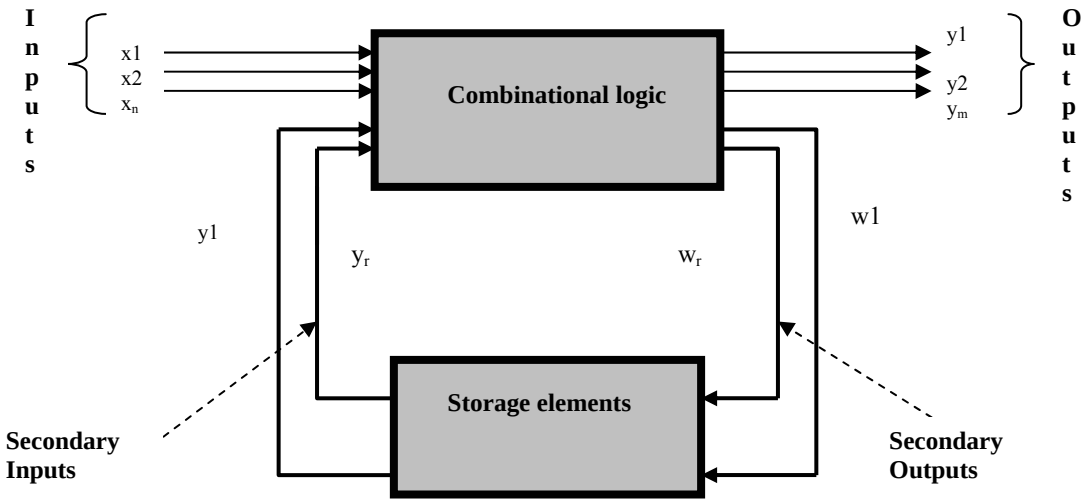


Figure 12: Model for sequential logic networks where combinational logic gates and memory elements are separated into two halves

The basic difficulty with testing the circuit model of the figure 12 is that the logic values y_1 to y_s are generally unknown. If they were observable then the testing of the top half of the circuit would be entirely combinational, and testing of the bottom half would be by monitoring y_1 to y_s during an appropriate set of primary input test vectors. However, assuming y_1 to y_s are inaccessible, developments as follows must be considered.

The first problem is to initialize the memory elements to a known state before any meaningful test procedure can .This may be done in the following ways:

i. Apply for a synchronizing input sequence to the primary inputs which force a fault-free circuit to one specific state irrespective of its initial starting state. This is only possible in special cases where the states always home to a given state and remain there in the presence of a certain input vector- a counter which free runs but is stopped when it reaches 1111… by a certain input vector is an example.

ii. Apply a homing input sequence to the primary inputs which finally gives a recognizable output sequence from a fault-free circuit irrespective of its initial starting state. Depending upon which output sequence is recognized, the final state of the circuit is known

iii. A special case of the homing input sequence is the distinguishing input sequence, where the circuit output response is different from every possible starting state. If this is possible, then both the initial and final states of the circuit are known. Most sequential circuits, however, do not have a distinguishing input sequence, although all do have one or more homing input sequences.

iv. Finally, a separate asynchronous reset may be applied to all internal latches and flip-flops to reset them to the 000…00 state ,or they may all be reset or set on the application of a specific input test vector which the circuit designer has built into the specific circuit.

With the circuit initialized to a known starting state the primary outputs maybe checked for correctness with appropriate primary test input vectors if necessary. If this test faults, the circuit is faulty and no further testing need be continued unless some diagnostic information is also sought. It also has been suggested that the successful completion of homing sequence, particularly if a long sequence, is itself a good test for the complete circuit, any deviation indicating some failure in the combinational or sequential elements. However, it is difficult to determine the fault cover that a homing test provides, and additional testing from the starting point of the initialized circuit is usually required.

In spite of research and development activities it here that viable ATPG programs are not available, use of stuck-at model becoming difficult because of feedback complexities. Some efforts have been made to partition the sequential circuit into an interactive cascade of one-state circuits, effectively spreading out the synchronous machine linearly in time instead of going around the one circuit model on each clock pulse, but unfortunately this introduces the equally difficult problem of having to model multiple stuck-at combinational faults.

In short Sequential automatic test pattern generation is a difficult problem. The many challenges we face in this area include reduction in the time and memory required to generate the tests, reduction in the number of cycles need to apply the tests to the circuit, and obtaining a high fault coverage. Adding to the complexity of this problem is that, unlike a combinational circuit where an untestable fault is also redundant, an untestable fault is not necessarily redundant in a sequential circuit.

3.9.1 Classification of test generation techniques for sequential Circuits

The taxonomy of various sequential test generation approaches is given in the figure 13[28]. Most of the approaches are limited to test generation for synchronous sequential circuits which may have some asynchronous reset /clear circuitry. However, some advances have also been made in test generation for asynchronous sequential circuits. Under the synchronous category, the three main approaches are based on the state table, gate-level circuits, and both register–transfer level (RTL) and gate-level circuits. The state table based approach is only useful for pure controllers for which a state is either available or easily extractable from its lower-level description. The gate-level approach can itself be divided up into topological analysis based, simulation based, and hybrid. The topological analysis based approach uses an iterative array model of the sequential circuit. Most methods in this category assume that the initial state of the sequential circuit is unknown, while a few others assume that the initial state is known to avoid the problem of initializing the memory elements. The topological analysis based approach assuming an unknown initial state is by far the most common approach among all sequential test generation approaches.

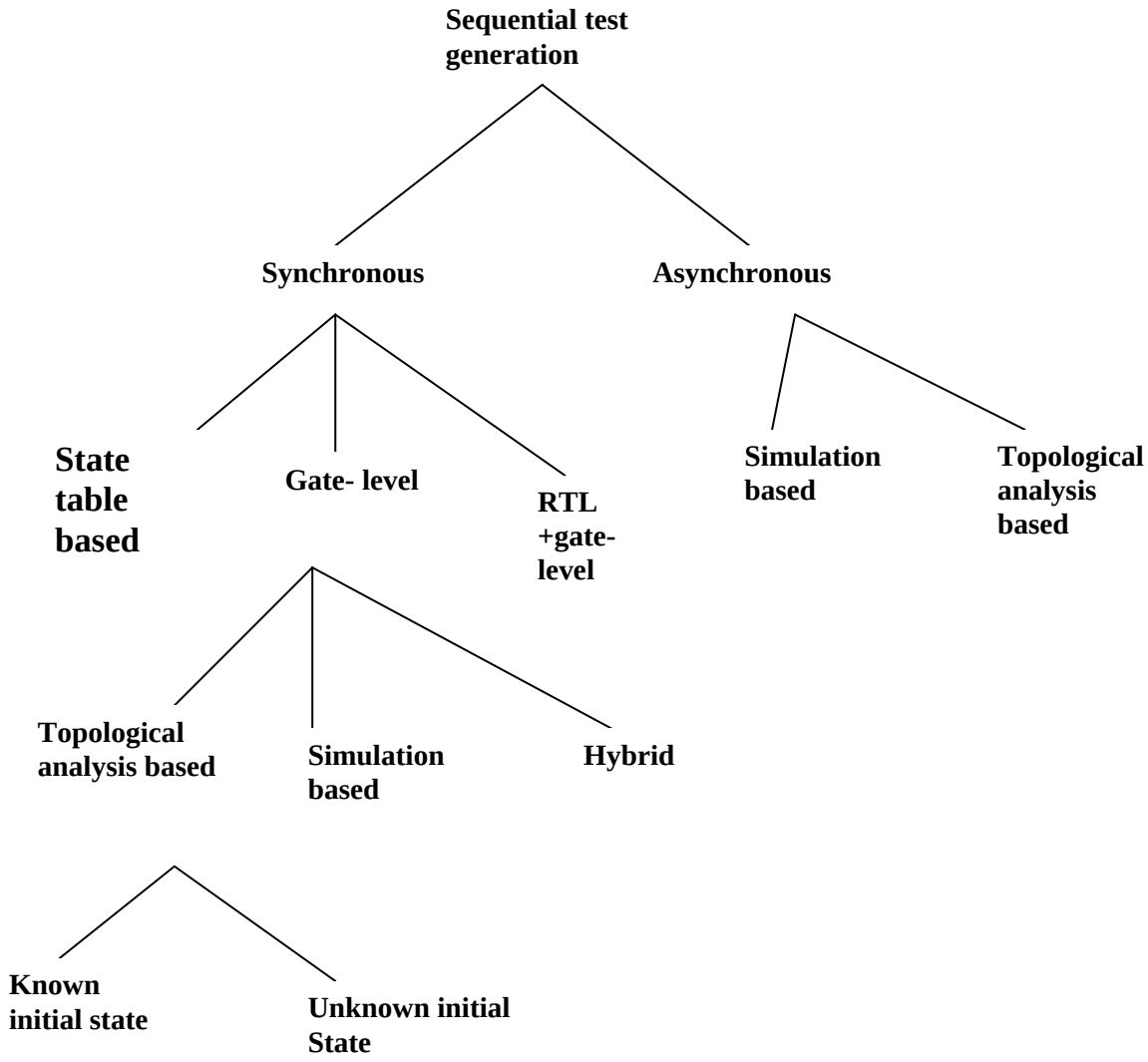


Figure 13: Taxonomy of sequential test generation

The simulation based approach uses an enhanced logic or fault simulator. The hybrid approach combines the topological analysis based and simulation based approaches. When the RTL description of the circuit is available in addition to its gate-level one, test generation can be significantly sped up since it becomes easier to justify a state and propagate the error to an observable output.

For asynchronous circuits, the two main approaches are simulation-based and topological analysis based. In the first approach, a potential test sequence is first derived by ignoring the circuit delays, and then simulated using the appropriate delay models to make sure it does not get invalidated by races, hazards and oscillations. If invalidation does occur, another potential test sequence is derived and the process is repeated. In the second approach, synchronous topological analysis based test generation is adapted to asynchronous circuits by only allowing the use of state tables.

3.10 Design for Testability

Design for testability (DFT) can loosely be defined as changes to a given circuit design that help decrease the overall difficulty of testing. The changes to the design typically involve addition or modification of circuitry such that one or more new modes of circuit operation are provided. Each new mode is called a test mode in which the circuit is configured only for testing. During the normal use, the circuit is configured on the normal mode and has identical input-output logic behavior as the original circuit design. However, the timing of the circuit may be affected by the presence of the DFT circuitry.

The key objective of DFT is to reduce the difficulty of testing. ATPG techniques for sequential circuits are expensive and often fail to achieve high fault coverage. The main difficulty arises from the fact that the state inputs and sate outputs cannot be directly controlled and observed, respectively. Controllability is measure of how easily a node in a circuit can be set to logic 0 and logic 1 by signals applied to the accessible (primary) input. Observability is a measure how easily the state of a given node (logic 0 or logic 1) can be determined from the logic signals at the accessible outputs. In such cases, a circuit may be modified using the scan design methodology, which creates one or more modes of operation which can be used to control and observe the values at some or all flip-flops.

This can help reduce the cost of test development. In fact, in many cases use of some scan is the only way by which the desired fault coverage target may be achieved. However, in most cases, the use of scan can increase the test application time and hence the test application cost. In a slightly different scenario, DFT circuitry maybe used to control and observe values at selected lines within the circuit using test points so as to reduce the number of test vectors required. In such a scenario, use of DFT may help reduce the test application cost.

3.10.1 Trade-offs

Most DFT techniques deal with either the resynthesis of an existing design or the addition of extra hardware to the design. Most approaches require circuit modifications and affect such factors as area, I/O pins and circuit delay. The values of these attributes usually increase when DFT techniques are employed. Hence, a critical balance exits between the amount of DFT to use and the gain achieved [29]. Test engineers and design engineers usually disagree about the amount of DFT hardware to include in a design.

Increasing area or logic complexity in a VLSI chip will result in increased power consumption and decreased yield. Since testing deals with identifying faulty chips, and decreasing yield leads to an increase in number of faulty chips produced, a careful balance must be reached between adding logic for DFT and yield. Normally yield decreases linearly as chip area increases. If additional hardware required to support DFT does not lead to an appreciable increase on fault coverage, then the defect level

will increase. In general, DFT is used to reduce test generation costs, enhance the quality of tests, and hence reduce defect levels. It can also affect test length, tester memory, and test application time. By employing structured DFT techniques test development time can be reduced. Without DFT, tests may have to be generated manually; with DFT they can be generated automatically. For design engineering organizations and individual the perception of DFT is generally negative:

- It adds work complication to the design methodology, adds tasks and risks to design schedule
- It negatively impacts design budgets as
  - power
  - area
  - timing
  - package pin requirements

However, for test professionals, the perceptions of DFT are usually positive and include such items as:

- having the ability to measure the quality level deterministically
- making it easier to generate the necessary vectors
- making it possible to support all test environments easily
  - wafer probe
  - manufacturing test
  - burn-in
  - life-cycle
  - board-level integration
  - engineering debug
  - customer return debug
  - process characterization
  - yield enhancement and failure analysis
- allowing the cost-of-test to be reduced in all environments
  - reduce tester complexity and cost
  - reduce tester time
  - reduce tester requirements (pins, memory depth, pin timing)

3.10.2 Classification of DFT techniques

DFT techniques normally fall into three categories, namely:

i. ad hoc design methods;
ii. structured design methods;
iii. self test

The first two of these methods usually require the use of some external comprehensive test facility, but the third method usually minimizes to a considerable extent the use of external test resources.

3.10.2.1 Ad-hoc methods

Ad-hoc methods target difficult-to-test parts of the Design under test (DUT) in order to improve controllability and observability of such a circuitry. Gate inputs and outputs, which are normally out of control or out of observation, are made accessible by adding a test point .Improving the controllability of internal nodes is mostly arranged by gate-based test points – the internal signal is enabled or disabled by control wire. These cells are not the only additional hardware, which is added to the original design (Figure 14).

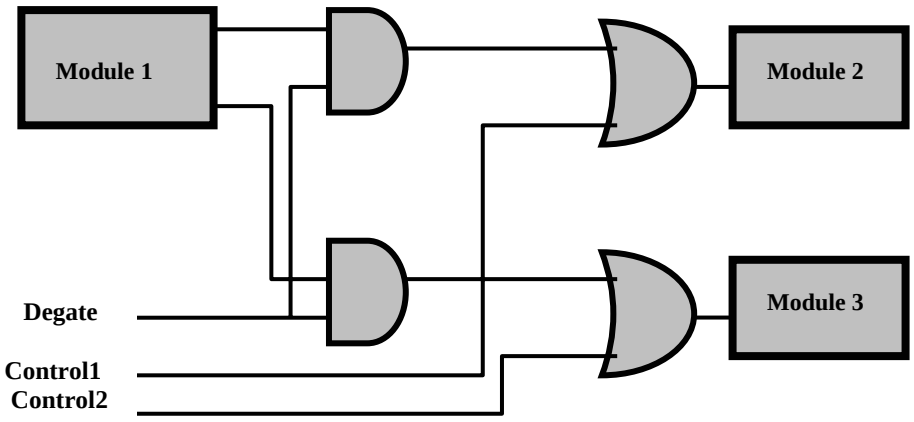


Figure 14: Module degating

The ad-hoc methods can be used on critical timing paths only very carefully to eliminate performance penalties caused for an example by test point insertion, because every gate causes some time delay. Ad-hoc solutions applied on case-by-case basis can yield significant increase of fault coverage, along with reductions in test time. The down side of an ad-hoc DFT method is the number of additional I/O pins required for the test mode control and the fact that we determine the best place to insert test point on case-by-case basis. There are several CAD tools which are able to assist in test point insertion, but ad-hoc DFT process is usually of manual test point insertion and can take considerable time. This process consists usually of manual test point insertion, developing test patterns, running fault simulations, evaluating fault coverage and repeating these steps until acceptable fault coverage is obtained. Typical targets for test point insertion are feedback loops, large counters, embedded core logic, asynchronous logic embedded clock generators, memory initialization inputs, intentional redundant logic, etc.

The ad-hoc DFT methods are based on good design practices learned from experience; some of them are summarized below [29]:

- Reset dependency during scan must be avoided, reset must be fast and easy , the reset circuitry must also be testable ; reset wire must be accessible directly from the chip pin and it is not recommended to gate this signal with some internal, nondeterministic logic;

- Long counters should be broken for the testing, also state machines with many states should be avoided or split to a group of smaller state machines; to test a long counter in all of the states takes a lot of clock cycles. Parallel test of separated parts takes a small fraction of this time;
- Do not use non-gate-level logic, or strictly separate the parts with different logic , or separate analog and digital parts; most ATPG tools are based on gate-level logic, such a circuitry may cause loss of controllability or slows down the ATPG processing;
- Do not use either data as clock or clock as data, there is a big possibility of skew effect or of some delay faults for example in a long ripple-counter or similar circuitry;
- Avoid gates with large number of an-in signals; a big set of test pattern is needed for full-scan;
- Provide test control for difficult-to-control signals;
- Do not use both edges of a clock signal;
- Do not use internal nodes that support other logic than 1/0; it is hard to decide whether the response is correct or no;
- There should not be any dynamic logic inside the DUT (pull-up, pre-charge ….); such logic is expected only on pins. If an analog circuitry is used then it must be isolated from other parts;
- Do not use any logic to select the test mode; test mode signal may be generated only by an external logic, i.e. through a chip pin. The Built-In Self Test is an exception to this rules;
- Floating state in 3-state buffers should be disabled during testing to evaluate the responses correctly we need pure logic values;
- Avoid the redundant logic;
- Hold the distance from the technology limits; using the chip on these limits (frequency, voltage etc.) is dangerous, this increases the risk of damaging the chip.
- Never build in logical redundancy unless it is specifically required ; by definition such redundancy can never be directly checked from the normal I/O pins;
- Make a list of key nodes during the design phase, and check that they can be easily accessed;
- If possible use level sensitive flip-flops in preference to edge-triggered types, since the behavior of circuits using the former is more secure than circuits using the latter.
- At all cost avoid the use of on-chip monostable circuits; these must be off-chip if really necessary. On-chip use some form of clocked counter-timer circuit;
- If at all possible avoid the use of asynchronously operating macros; make all sequential circuits operate under rigid clock control unless absolutely impossible;
- Ensure that all storage elements, not only in counters but in PLAs and random-access memory can be initialized before test; this also applies to any internal cross-coupled NANDS and NORs which act as local latches;

- Limit the fan-out from individual gates as far as possible so as not to degrade performance and to ease the task of manual or automatic test pattern generation;
- Provide all means to break all feedback connections from one partition of a circuit to another partition (global feedback)so that each partition may be independently tested;
- Consider the advantage of using separate test clocks to ease certain sequential checks;
- Avoid designing clever combinational modules which perform different duties under different circumstance ; in general design a module as simply as possible to do on job;
- Keep analogue and digital circuits on chip physically as far apart as possible , with completely separate or very decoupled d.c supply connection;
- When designing a complex VLSI custom IC ensure that the vendor can supply details of appropriate vectors for test of larges macros such as PLAs, memory, etc;
- Consider adding some online parity or other check bits if fault detection or fault correction would bee advantageous;
- Remember that a vendor's 100% toggle test on internal nodes of a custom IC does not guarantee a fully fault-free circuit;
- Some commercial IC testers may not cater for don't care conditions and may require to know whether the actual design should give a logic 0 or a logic 1 under test conditions;

The major DFT guidelines may be summarized as follows:

- Maximize the controllability and observability of all parts of the circuit or system by partitioning or other means;
- Provide means of initializing all internal latches and flip-flops at the beginning of any test;
- Keep analogues and digital circuits physically and electrically as separate as possible;
- Avoid asynchronous working and redundancy if at all possible;

3.10.2.2 Scan design techniques

Scan testing is defined as the process of using scan architecture to conduct testing. Scan is known as a structured methodology, because it can be standardized, is repeatable, and is easily automatable (both in insertion and in vector generation).A scan architecture allows a data state to be placed within the chip by using scan shift registers, and also allows the data state of a chip to be observed by using those same scan shift registers. Scan architecture also allows algorithmic software tools to verify the test design's correctness and to create or generate the necessary structural test vectors required to verify that the rest of the chip has passed "defect free" through the manufacturing process. Multiple scan shift registers, or scan chains, help to optimize the vector depth required by the tester. Overall, a scan testing methodology can enhance the ability to achieve a high-quality metric, reduce the cost-of-test, reduce the

time it takes to get the vectors to the tester – and therefore, to get the chip into volume production. The testing procedure for the circuit is as follows [27]:

i. The circuit is switched from its normal mode to scan mode, which converts the storage elements into scan-path shift register;
ii. The switching and storage action of this shift register is first checked by clocking through a pattern of 0s and 1s under the control of the test clock;
iii. If this primary initial test is all correct, an input test vector is applied to the primary inputs, and a chosen pattern of 0s and 1s is serially loaded in to the shift register under the control of the test clock, the latter becoming the secondary inputs to the combinational logic,
iv. The circuit is switched back to its normal mode, and the clock operated once so as to latch the resultant secondary outputs from the combinational logic back into the shift register;
v. 'The circuit is switched back to its test mode, and the test clock operated so as to scan out the latched data from the shift register to the scan-out I/O for checking;
   Steps 3, 4, 5 are repeated as many times as necessary in order to test all the combinational logic circuits.

3.10.2.2.1 Scan cell designs

The fundamental cell in scan design is a scan –cell. This cell is an independently accessible unit of scan circuitry serving as a control point and observation point for ATPG. There are three widely used scan cell designs [20]:

i. Muxed-D Scan cell

The D storage element is one of the most widely used storage elements in logic design. Its basic function is to pass a logic value from its input to its output when a clock is applied. A D flip-flop is an edge-triggered D storage element, and a D latch is a level-sensitive D storage element. The most widely used scan cell replacement for the D storage element is the muxed-D scan cell. Figure 15 shows an edge-triggered muxed-D scan cell design. This scan cell is composed of a D flip-flop and a multiplexer. The multiplexer uses a scan enable (SE) input to select between data input (DI) and the scan input (SI).

In normal/capture mode, SE is set to 0.The value present at the data input DI is captured into the internal D flip-flop when a rising clock edge is applied. In shift mode, SE is set to 1. The SI is now used to shift new data to D flip-flop while the content of the D flip-flop is being shifted out.

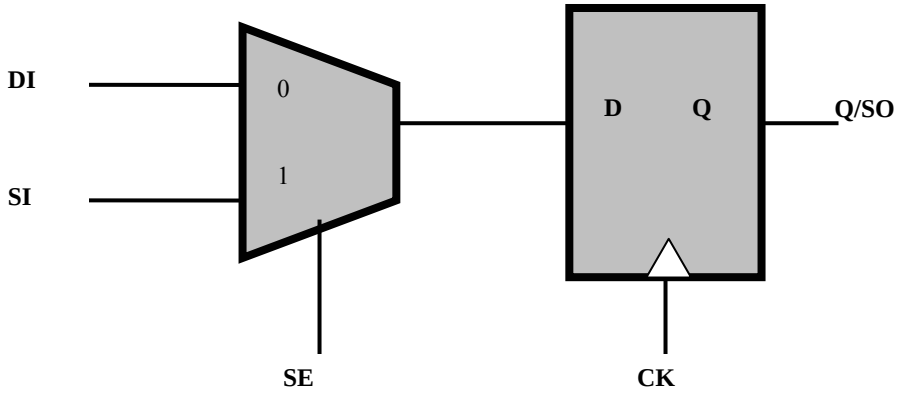


Figure 15: Edge triggered muxed D scan cell

Major advantages of using a muxed-D scan cells are their compatibility to modern designs using single-clock D flip-flops, and the comprehensive support provided by existing design automation tools. The disadvantage is that each muxed-D scan cell adds a multiplexer delay to the functional path.

ii. Clocked-Scan Cell

An edge-triggered clocked scan-cell can also be used to replace a D flip-flop in a scan design. Similar to a muxed-D scan cell, a clocked-scan cell also has data input DI and a scan input SI; however, in clocked-scan cell, input selection is conducted using two independent clocks, data clock DCK and shift clock SCK as shown in figure 16. In normal/capture mode, the data clock DCK is used to capture the value present at the data input DI into the clocked scan-cell. In shift mode, the shift clock SCK is used to shift new data from the scan input SI into the clocked–scan cell, while the current content of the clocked scan cell is shifted out.

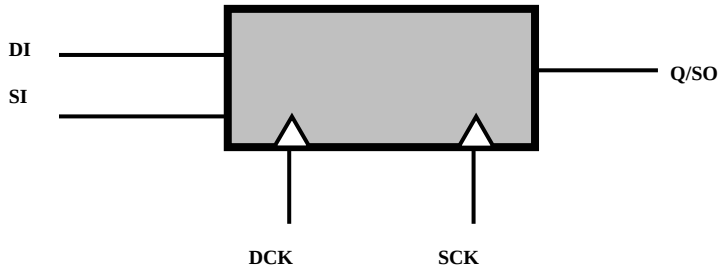


Figure 16: Clocked scan cell

The major advantage of using this structure is that it results in no performance degradation on the data input.The major disadvantage however, is that it requires additional shift clock routing.

iii. LSSD Scan cell

While muxed-D scan cells and clocked-scan cells are generally used for edge triggered, flip-flop-based designs, an LSSD scan cell is used for level-sensitive, latch-based designs. The scan cell contains two latches, a master two-port D latch and a slave D latch. Three clocks are used to select between the data input and scan input to drive the latches. Either of the two latches can be used to drive the combinational logic of the design. In order to guarantee race-free operation, the three clocks are applied in a non overlapping manner.

The major advantage of using an LSSD scan cell is that it allows us to insert scan into a latch-based design. In addition, designs using LSSD are guaranteed to be race-free. The major disadvantage, however, is that the technique requires routing for the additional clocks, which increases routing complexity.

3.10.2.2.2 Scan architectures

In this section we describe the three popular scan architectures.

i. Full-Scan Design

In full-scan design, all the storage elements are replaced with scan cells, which are then configured as one or more shift registers (also called scan chains) during the shift operation. As a result, all the inputs to the combinational logic, including those driven by scan cells, can be controlled and all outputs from the combinational logic, including those driving scan cells, can be observed. The main advantage of full-scan design is that it converts the difficult problem of sequential ATPG into the simpler problem of combinational ATPG.

ii. Partial-Scan Design

Unlike full-scan design where all storage elements in a circuit are replaced with scan cells, partial-scan design only requires that a subset of storage elements be replaced with scan cells and connected into scan chains. Partial-scan design was used in the industry long before full-scan design became the dominant scan architecture. In order to reduce the test generation complexity, many approaches have been proposed for determining the subset of storage elements for scan cell replacement. Scan cell selection can be conducted by using a functional partitioning approach (partitioning data portion and control portion), feed-forward partial scan design (storage elements to be replaced by scan cells are selected to make the sequential circuit feedback free so that test generation complexity is reduced and silicon overhead area is low).

iii. Random-Access Scan Design

Full-scan design and partial-scan design can be classified as serial scan design, as test pattern application and test response acquisition are both conducted serially through scan chains. The major advantage of serial scan design is its low routing overhead, as scan data is shifted through adjacent scan cells .Its major disadvantage, however, is that individual scan cells cannot be controlled or observed without affecting values of other scan cells within the same scan chain. High switching activities at scan cells can cause excessive test power dissipation resulting in circuit damage, low reliability or even test-induced

yield loss. Random-access scan attempts to alleviate these problems by making each scan cell randomly and uniquely addressable, similar to storage cells in random-access memory.

3.10.2.3 Boundary scan

Boundary scan is a method for testing interconnects (wire lines) on printed circuit boards or sub-blocks inside an integrated circuit. Boundary scan is also widely used as a debugging method to watch integrated circuit pin states, measure voltage, or analyze sub-blocks inside an integrated circuit [29].

The Joint Test Action Group (JTAG) developed a specification for boundary scan testing that was standardized in 1990 as the IEEE Std. 1149.1-1990. In 1994, a supplement that contains a description of the Boundary Scan Description Language (BSDL) was added which describes the boundary-scan logic content of IEEE Std 1149.1 compliant devices. Since then, this standard has been adopted by electronic device companies all over the world. Boundary scan is nowadays mostly synonymous with JTAG. The boundary scan architecture provides a means to test interconnects and clusters of logic, memories etc. without using physical test probe. It adds one or more so called 'test cells' connected to each pin of the device that can selectively override the functionality of that pin. These cells can be programmed via the JTAG scan chain to drive a signal onto a pin and across an individual trace on the board. The cell at the destination of the board trace can then be programmed to read the value at the pin, verifying the board trace properly connects the two pins. If the trace is shorted to another signal or if the trace has been cut, the correct signal value will not show up at the destination pin, and the board will be known to have a fault. When performing boundary scan inside integrated circuits, cells are added between logical design blocks in order to be able to control them in the same manner as if they were physically independent circuits. For normal operation, the added boundary scan latch cells are set so that they have no effect on the circuit, and are therefore effectively invisible. However, when the circuit is set into a test mode, the latches enable a data stream to be passed from one latch to the next. Once the complete data word has been passed into the circuit under test, it can be latched into place. As the cells can be used to force data into the board, they can set up test conditions. The relevant states can then be fed back into the test system by clocking the data word back so that it can be analyzed. By adopting this technique, it is possible for a test system to gain test access to a board. As most of today’s boards are very densely populated with components and tracks, it is very difficult for test systems to access the relevant areas of the board to enable them to test the board. Boundary scan makes this possible.

3.10.2.3 Built-in self test

With recent advances in semiconductor manufacturing technology, the production and using of VLSI circuits has run into a variety of testing challenges during wafer probe, wafer sort, pre-shipping screening, incoming test of chips and boards, test of assembled boards, system test, periodic maintenance ,repair test, etc. Traditional

techniques that use to ATPG software to target single faults for digital circuit testing have become quite expensive and can no longer provide sufficiently high fault coverage for deep submicron or nanometer designs from the chip level to the board and system levels.

One method to alleviate these testing problems is to incorporate built-in self-test (BIST) features into a digital circuit at the design stage. With BIST, circuits that generate test patterns and analyze the output response of the functional circuitry are embedded in the chip or elsewhere on the same board where the chip resides.
The important advantages of BIST are [29]:

- Easy access to circuit under test (CUT) even in highly complex VLSI circuits where number of external pins is limited;
- No necessity to store the input and output test patterns;
- Reducing test development time;
- At-speed testing;
- Reduced manufacturing test time, reduced time to market
- Since BIST implements most of the test functions on-chip ,the origin of errors can be easily traced back to the chip;

The disadvantages are:

- Increased overhead and performance penalties
- Additional design effort
- Additional risk to project (BIST could cause yield loss to its circuitry)

There are two general categories of BIST technique for testing random logic :(1) online BIST and (2) offline BIST.

Online BIST is performed when the functional circuitry is in normal operational mode. It can be done either concurrently or nonconcurrently. In concurrent online BIST, testing is conducted simultaneously during normal functional operation. The functional circuitry is usually implemented with coding techniques or with duplication and comparison. When an intermittent or transient error is detected, the system will correct the error on the spot, rollback to its previously stored system states, and repeat the operation, or generate an interrupt signal for repeated failures. In nonconcurrent online BIST, testing is performed when the functional circuitry is in idle mode. This is often accomplished by executing diagnosis software routines (macrocode) or diagnosis firmware routines (microcode). The test process can be interrupted at any time so that normal operation can resume.

Offline BIST is performed when the functional circuitry is not in normal mode. This technique does not detect any real-time errors but is widely used in the industry for testing the functional circuitry at the system, board, or chip level to ensure product quality.

Functional offline BIST performs a test based on the functional specification of the functional circuitry and often employs a functional or high-level fault model. Normally such a test is implemented as diagnostic software of firmware.

Structural offline BIST performs a test based on the structures of the functional circuitry. There are two general classes of structural offline BEST techniques: (1) external BIST, in which test pattern generation and output response analysis is done by circuitry that is separate from the functional circuitry being tested, and (2) internal BIST, in which the functional storage elements are converted into test pattern generators and output response analyzers. Some external BIST schemes test sequential logic directly by applying test patterns at the input s and analyzing the responses at its outputs. Such techniques are often used for board-level and system-level self-test.

Figure 17 shows a typical BIST system using the structural offline BIST technique. The test pattern generator (TPG) automatically generates test patterns for application to the inputs of the circuit under test (CUT).

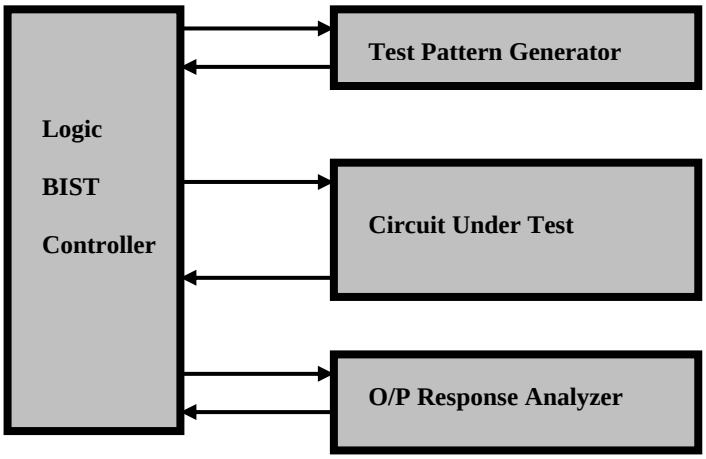


Figure 17: A typical logic BIST System

The output response analyzer (ORA) automatically compacts the output responses of the CUT into a signature. Specific BIST timing control signals, including scan enable signals and clocks, are generated by the BIST controller for coordinating the BIST operation among the TPG, CUT, and ORA. The BIST controller provides a pass/fail indication once the BIST operation is complete. It includes comparison logic to compare the final signature with an embedded golden signature, and often comprises diagnostic logic for fault diagnosis. As compaction is commonly used for output response analysis, it is required that all storage elements in the TPG, CUT, and ORA are initialized to known states prior to self-test, and no unknown values be allowed to propagate from the CUT to the ORA.

3.10.2.4.1 Test pattern generation using LFSR in BIST

Several types of test pattern generators are used in BIST. The most important generator used in BIST is Linear Feedback Shift Register (LFSR). It is more area efficient than a binary counter, requires less combinational logic per flip-flop, can

usually work at higher clock frequency, and the LFSR output sequence can be considered to be pseudorandom. Let us suppose that we have an n bit LFSR. It is not possible to generate all the 2^n different patterns after seeding with one seed only. This can be considered to be a disadvantage of using an LFSR The LFSR without external input is sometimes called autonomous LFSR (ALFSR) and is suitable for test pattern generation.

Two basic types are used in practical applications: internal feedback (Type 1, Figure 18) and external feedback (Type 2, Figure 19) .Both implementations require the same amount of logic in terms of exclusive-OR gates.

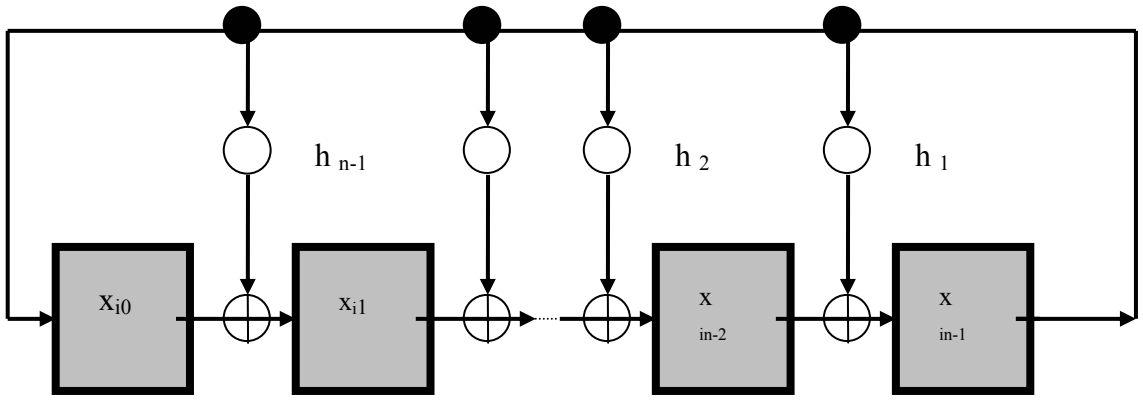


Figure 18: Internal feedback LFSR

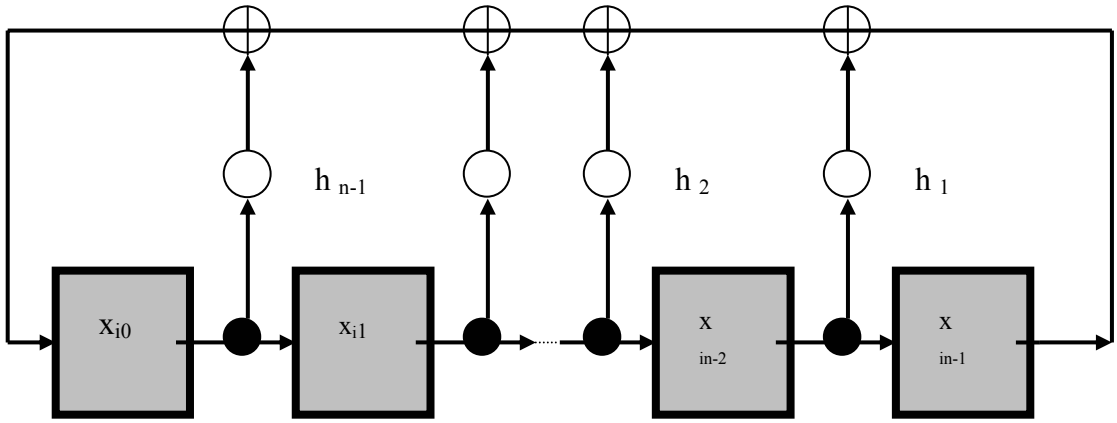


Figure 19: External feedback LFSR

The XOR gates are connected according to the characteristic polynomial f(x). This polynomial can be described as:

$$f(x) = \sum_{i=0}^{n} h_i x^i$$

Where h_i is the feedback tap coefficient. If h_i =1, feedback tap exists, for h_i =0 no feedback exists. An n bit LFSR has the coefficients h_n and h_0 equal to 1 as they guarantee concatenation of the main loop. The number of XOR gates in the ALFSR is equal to the number of non-zero coefficients of the code generating polynomial deducted by 2. On the other hand, type 1 LFSR has, at most, one XOR gate in any path between flip-flops. Both types of the LFSR perform a division of the input polynomial by the characteristic polynomial in the binary Galois field (finite field). The quotient can be read at the LFSR output. The remainder of the division can be found in the internal feedback LFSR flip-flops. If a LFSR is supposed to run in autonomous mode only, then the input XOR gate is not needed. The relationship between the register state at a present time and the register state after the clock cycle can be described by a matrix equation Q` = AQ, where Q is the current state, Q` is a resulting state and A is the transformation matrix, The transformation matrix coefficients represents the LFSR tap topology.

An autonomous LFSR generates the code words of cyclic linear codes on all flip-flop outputs. The code words can be characterized by the code word length p and by the number of the information bit n. The number of LFSR flip-flops is equal to n. If the LFSR output is concatenated with a (p-n) bit shift register, then 2^n different code words can be generated. Each code has n information bits (stored in the LFSR) and (p-n) check bits (stored in the shift register). The information bits can define the code words unambiguously. As the code is cyclic a new word must be obtained by cyclic shifting of a code word. From this fact it follows that automation formed by the LFSR and the shift register could be replaced by a p bit shift register with a feedback tap from the last to the first flip-flop and if seeded with a code word the generated code word is the same as that obtained by the LFSR with a shift register.

The characteristic LFSR polynomial corresponds to the parity check polynomial of the generated code, which means that if we create the product of a code polynomial and the check polynomial we obtain a zero polynomial. The code is also characterized by the generator polynomial of the degree n .The generator polynomial is unique within the code polynomials and each code polynomial is a product of it and some other polynomial .The relationship between the generator polynomial and the parity check polynomial is then given by the equation :

$$h(x)g(x) = X^p - 1$$

The generator polynomial maybe:

- Irreducible –it is not possible to factorize the polynomial;

- Reducible – it can be factored into a product of simpler irreducible polynomials;
- Primitive – this polynomial must be irreducible and it divides the polynomial $x2^{exp(n)-1}$ -1 and it doe not divide any polynomial of a lower degree;

It is computationally difficult to decide whether a polynomial is primitive but the primitive polynomials are tabulated for all n up to sufficiently high values.

It is possible to use the generator polynomial as a characteristic LFSR polynomial in which case the LFSR generates code words called dual code.
The internal states of both types of the LFSR are periodically repeated when the LFSR runs in the autonomous mode. The length of the longest possible sequence of unique patterns for any kind of an n bit LFSR is equal to 2^n -1.The longest sequence can be obtained when connecting the LFSR feedback according to a characteristic polynomial that is a primitive generator polynomial of the dual code. To obtain the periodical LFSR behavior the LFSR has to be seeded with a non-all-zero state. Any LFSR initialized to the all 0s state remains in this state; the sequence length is equal to 1. In some situations it is not desirable to use primitive generator polynomials. Irreducible or reducible polynomials with high code distance of the dual code are used which results in better quality pseudoexhaustive test set. In this case the length of the obtained sequence is a fraction of the maximum sequence length and the LFSR has to be reseeded several times. It is possible to modify the LFSR in order to produce the all zero pattern within the primitive polynomial sequence. To do this we have to add a decoding logic to the LFSR, which selects one LFSR state and forces the LFSR to reach the all-zero state in the next clock cycle. This way of completing LFSR sequence is not advantageous because of the additional hardware overhead and worse dynamical parameters of the LFSR .If we want to prolong the LFSR period, it is more advantageous to use a LFSR with one more flip-flop.

3.10.2.4.2 Output response analysis

Storage of fault dictionary (all test inputs with correct output responses on chip requires too much memory to be a practical technique). The simplest practical method for analyzing the output response is to match the outputs of two identical circuits. Identical circuits may be available either because the function being designed naturally leads to replicated sub functions or because the functional circuitry is duplicated redundantly for concurrent checking. If identical outputs are not available it is necessary to resort to some technique for compaction of the test response. Techniques for reducing the volume of the output data were originally developed in connection with portable testers. Their use is usually called compact testing, but this technique is sometimes also called response compression. In compact testing, the output response pattern is passed through a circuit, called a compactor that has fewer output bits than input bits. The output of the compacter is called the signature of the test response. The aim is to reduce the number of bits that must be examined to determine whether the circuit under test in faulty.

The choice of compaction technique is influenced mainly by two factors: (1) the amount of circuitry required to implement the technique, and (2) the loss of effective fault coverage. In general a fault will go undetected if none of the input test patterns produces an incorrect circuit output in the presence of the fault. With output response compaction it is also possible for a fault to fail to be detected even though the output response differs from the fault-free response. This will happen whenever the output response from a faulty circuit produces a signature that is identical to the signature of a fault-free circuit. This phenomenon is called aliasing.

Signature analysis

The most popular BIST compaction circuit is an LFSR with its input equal to the output response of the circuit under test. This circuit was called a cyclic code checker when it was first proposed. This method of output response compaction is most often called signature analysis. The term "signature" describes the LFSR contents after shifting in the response pattern of the circuit being tested.

The usefulness of signature analysis depends on the fact that the final values of the LFSR flip-flops, the signature, depend on the bit pattern that is applied at the input. If a fault causes the output bit sequence to change; this will usually result in a different signature in the LFSR. However aliasing can occur. It is possible for a fault to cause an output bit sequence that produces the same final LFSR contents as the fault-free circuit. In this case the fault will be undetected. The output sequences that have this property depend on the structure of the LFSR used .They are characterized in terms of division of the Galois field polynomial representation of the LFSR and the output response sequences. Any compaction technique can cause some loss of effective fault coverage due to aliasing.

There are two methods for signature analysis for a multi-output circuit under test. One of them, the serial signature analyzer, uses a multiplexer to direct each of the outputs to the LFSR in turn. A circuit for this is shown in figure 20**.**

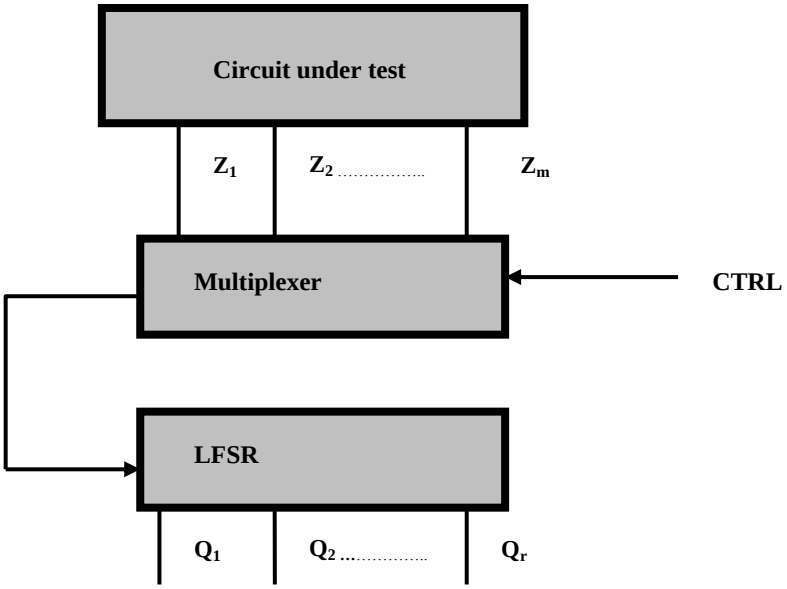


Figure 20: Serial signature analysis using a multiplexer

With this scheme the input test patterns would have to be applied to the network m times for an m-network.

The other technique, the parallel signature analyzer, compacts K network outputs in parallel using a K-bit parallel code checker figure 21.

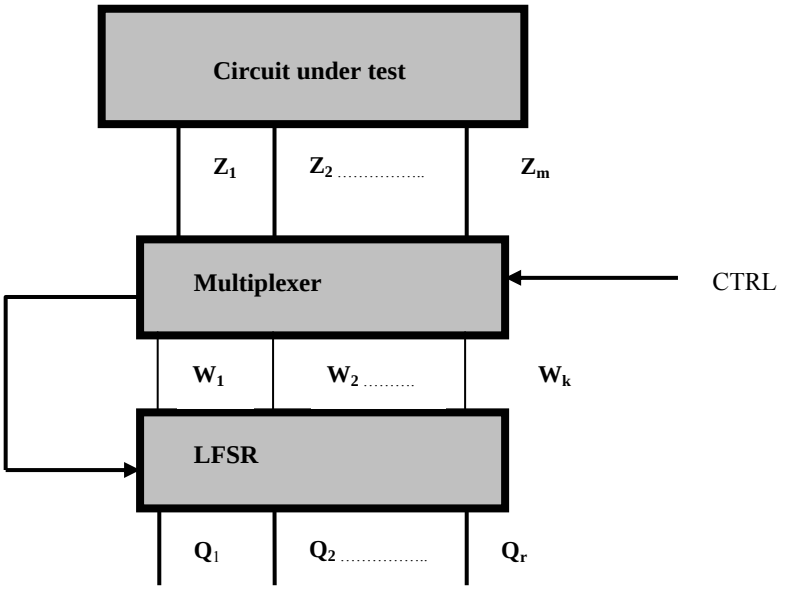


Figure 21: Parallel signature analysis

The parallel technique requires each test pattern to be applied only m/K times. In the parallel technique, network outputs are connected to the LFSR through XOR gates added to the shift lines between stages as well as connecting a network output to the first LFSR stage.

In general, the parallel signature analyzer is faster but requires more added circuitry than the serial signature analyzer. The parallel signature analyzer has an additional alias source. An error in the output Z_j at the time t_i followed by an error in output Z_{j+1} at time t_{i+1} will have no effect on the signature. More generally, an error in output Z_j at time ti followed by an error in output Z_{j+h} at time t_{i+h} will have no effect on the signatures.

3.10.2.4.3 BIST architectures

There are typically two basic BIST execution options: 1) Test-per-clock (Parallel BIST) and 2) Test-per-scan (Serial BIST)[30]. In Parallel BIST, test patterns are applied to the CUT by the test generator. The response analyzer captures the responses every clock cycle .In Serial BIST, test patterns are shifted into a serial scan path or multiple scan paths to test the CUT. The test responses are subsequently captured by the scan flip-flops and shifted out to the response analyzer while new patterns are being shifted in.

Serial BIST architectures

In this sub section some Serial BIST architectures are discussed. They are

i. **LOCST :** The Level sensitive scan design on-chip self-test (LOCST) architecture features two boundary scan registers used to buffer primary inputs and outputs. These registers are serially connected with the internal scan path, as well as with a test generator and a test-response compactor. The last two blocks are implemented by means of the LFSRs. The design includes an on-chip monitor acting as a BIST controller and an error detection circuitry employed to compare the final value in the compactor with a good signature. The BIST circuitry operates in such a way that the input vectors produced by the LFSRs are applied serially to the primary inputs and the internal nodes of the CUT through boundary scan and the internal scan flip-flops. Responses are captured by the internal scan and the second boundary scan register, and are subsequently shifted serially to the LFSR performing output data compaction. The content of the scan path can also be shifted out of the chip using an output lines and an input line is used to initialize the test generator.

ii. **CEBS:** The architecture of the Centralized and Embedded BIST with boundary scan (CEBS) is similar to that of LOCST. The only difference is in the location of the test generator and the test-response compactor. In contrast to the LOCST, these modules are implemented by means of first bits of the input scan register, and the last bits of the output scan register, respectively. Thus certain inputs of the CUT are

stimulated in parallel, while other are loaded serially. Also for some outputs, the test –response compactor is as a multiple input signature register (MISR), and for the remaining ones, it acts as an LFSR. Consequently, the scheme saves some silicon area and applies test vectors slightly faster than LOCST.

iii. **STUMPS:** The application of LOCST or CEBS requires a large number of clock cycles due to the inherent limitations of single scan chains. An attempt to overcome this constraint has been made in the Self-testing using MISR and parallel shift register sequence generator (STUMPS) architecture. In this approach the LFSR used as a test generator feeds a multiplicity of scan paths, while the serial outputs of the scan paths drive the MISR inputs. The use of multiple scan chains can significantly reduce the test application time. Since the scan paths may be of different lengths, every time a pattern is to be produced, the generator is run for c clock cycles, where c is the size of the longest scan chain. The resultant fault coverage is the stumps architecture may not be satisfactory due to the structure of the test generator. If the scan paths are fed directly from adjacent bits of the LFSR, then this close proximity will cause the neighboring scan chains to contain test pattern which are highly correlated. This phenomenon can adversely affect fault coverage, as patterns seen by the CUT will not be pseudo-random. Furthermore, the quality of the test can be deteriorated by the presence of linear dependencies in LFSR-generated sequences. In fact, the inability to produce some bit combinations may affect all LFSR – based applications of BIST, such as different test generation scenarios, reseeding of LFSRS, and others. In order to alleviate this problem, phase shifter are used when designing two-dimensional generators, and extra precautions have to be taken in selecting feedback polynomials .A typical phase shifter consists of XOR trees placed between the LFSR and the CUT in order to avoid shifted versions of the same data in various scan paths. Moreover, a preference should be given to feedback polynomials with a bigger number of feedback taps, since only these polynomials guarantee a probability of having linearly dependent bit positions in short test sequences at acceptable levels.

Parallel BIST Architectures

They can be classified as following [30]:

i. **BEST**: One of the first parallel BIST architectures proposed was the built-in evaluation and self-test (BEST) scheme. It essentially resembles a scheme used for board testing. The CUT inputs are driven by the generator of pseudo-random patterns, and the test responses are directed to a MISR. An additional external maintenance system has to supply the BEST scheme with a seed for the test generator, test length, and the expected signature. The extra four pins are used to facilitate

proper execution of BIST session. They include the following lines: test clock enable, test strobe, test data in, and test data out. The scheme requires extensive fault simulation to assess resultant fault coverage and might be ineffective for some categories of circuits.

ii. **BILBO:** The built-in logic block observer (BILBO) is one of the earliest structures designed specifically for test-per-clock BIST schemes .It combines the function of a register, shift register, LFSR, and MISR built around one set of latches. Therefore, each BILBO module can act as either generator or compactor, although in a given test session two of these blocks are required to test a module of the CUT, one assuming the role of generator, the other acting as a compactor. In the next test session these roles can be exchanged, and the content of a BILBO acting formerly as a compactor can be treated as a seed in a test generation mode. This scenario assumes, pessimistically, that the content of MISR cannot be considered as valuable pseudo-random vectors, but it has been proven that the MISR performing test response compaction can simultaneously act as a source of random patterns. This allows a designer to reduce the number of test phases, as the same BILBO module can be employed to observe as well as to simulate two different blocks of the CUT, or even, under special circumstances, a single BILBO can capture responses which are subsequently used as tests for the same CUT.

iii. **CSTP:** The circular self-test path (CSTP) BIST architecture converts some of the circuit flip-flops into self-test cells rather than using conventional LFSRs and MISRs. The cells are grouped into registers forming a circular shift register that simultaneously performs vector generation and response-data compaction. In the test mode, after placing all registers into a known state, the circuit operates for a number of clock cycles (registers that are not involved in the circular path work in the normal mode), and ,next, the entire signature, or its part left in the circular path, is scanned out for evaluation. Clearly this approach speeds up test application, as test responses do not have to be shifted out before the new vector is applied. The entire CUT can be treated as an FSM-based test generator with a nonlinear feedback function provided by the CUT itself.

iv. **Other Parallel BIST architectures:** Various other parallel BIST have been proposed. An approach similar to the CSTP is known as automated BIST. It selectively replaces memory elements of the CUT with a special BIST flip-flops and interconnects them in order to obtain a circular chain.

Self-test storage cells have also been used in the simultaneous self-test (SST) where a modified scan chain is employed to produce test vectors and collect test responses.

3.11 Fault Diagnosis

Diagnosis is the process of locating the faults in a structural model of the Unit under test (UUT). Diagnosis consists of locating the physical faults in a structural model of the Unit under test [31]. For some digital systems, each fabricated copy is diagnosed to identify the faults so as to make decisions about repair. This is the case for a system such as a printed circuit board, where pins identified as being fault can be replaced and the opens at or shorts between pins of a chip may be repaired via re-soldering. In such cases, diagnosis must be performed on each faulty copy of the system; hence, the cost of diagnosis should be low.

Digital VLSI chips are, by and large, un-repairable and faulty chips must be discarded. However, diagnosis is still performed on a sample of faulty chips, especially whenever chip yield is low or the performance of a large proportion of fabricated chips is unacceptable. In such a scenario, the objective of diagnosis is to identify the root cause behind the common failures or performance problems to provide insights on how to improve the chip yield and/or performance. The insights provided are used in variety of ways: (a) to change the design of the chip, (b) to change one or more design rules, (c) to change one or more steps of the design methodology, such as extraction, simulation, validation, and test generation, (d) to change the fabrication process, and so on. Relatively larger amounts of effort and resources may be expended on diagnosis in this scenario, since the accrued costs can be amortized over a large volume of chips produced. The ability to perform diagnosis on a sample of failing chip allows the cost of diagnosis to be viewed as fixed. This is unlike the other diagnosis scenario were each faulty copy of the system must be diagnosed to make decisions about repair.

A fabricated copy of the circuit can be diagnosed by applying tests to the inputs of the circuit, capturing the response at its outputs, and analyzing the captured response. This type of diagnosis is called logic diagnosis. Logic diagnosis is typically supplemented by more invasive physical diagnosis. If the CUT is already packaged, then the packaging is removed. In addition, layers of material may be removed at specific areas of the chip. VLSI features may then be examined using microscopes. Probes, such as electron-beam probes, may also be used to make measurements at internal circuit lines.

Physical diagnosis is expensive and destructive. Hence, logical diagnosis is typically used to first identify the likely faults. Physical diagnosis is then used to verify the results of logic diagnosis , a process that is sometimes called root-cause analysis .A CUT is said to be accurately diagnosed if the set of faults identified by logic diagnosis as the possible causes of CUT ‘s failure includes the real cause of failure. Diagnostic accuracy is defined by the proportion of all CUTs that are diagnosed accurately. Diagnostic resolution can be defined as the average number of faults that are identified by logic diagnosis as possible causes of CUT failure. The degree of the accuracy to which faults can be located is called diagnostic resolution. Functionally equivalent faults (FEF) cannot be distinguished. The partition of all faults into distinct

subsets of FEF defines maximal fault resolution .A test that achieves maximal fault resolution said to be a complete fault location test.

3.11.1 Combinational fault diagnosis methods

This approach does most of the work before the testing experiment. It uses fault simulation to determine the possible responses to a given test in the presence of faults. The database constructed in this step is called a fault table or a fault dictionary. To locate faults, one tries to match the actual results of test experiments with one of the pre computed expected results stored in the database. The result of the test experiment represents a combination of effects of the fault to each test pattern. That's why we call this approach combinational fault diagnosis method. If this look-up process is successful, the fault table (dictionary) indicates the corresponding fault(s).

3.11.2 Fault table

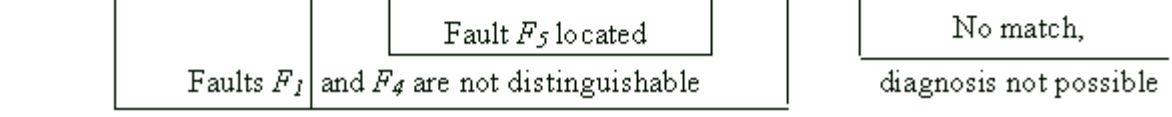
In general, a fault table is a matrix FT =$|a_{ij}|$ where columns F_j represent faults, rows T_i represent test patterns, and $a_{ij} = 1$ if the test pattern T_i detects the fault F_j, otherwise if the test pattern T_i does not detect the fault F_j, $a_{ij} = 0$. Denote the actual result of a given test pattern by 1 if it differs from the pre computed expected one, otherwise denote it by 0. The result of a test experiment is represented by a vector E =$|e_i|$ where $e_i = 1$ if the actual result of the test patterns does not match with the expected result, otherwise $e_i = 0$. Each column vector f_j corresponding to a fault F_j represents a possible result of the test experiment in the case of the fault F_j. Three cases are now possible depending on the quality of the test patterns used for carrying out the test experiment [29]:

1. The test result E matches with a single column vector f_j in FT. This result corresponds to the case where a single fault F_j has been located. In other words, the maximum diagnostic resolution has been obtained.
2. The test result E matches with a subset of column vectors $\{f_i, f_j \ldots f_k\}$ in FT. This result corresponds to the case where a subset of indistinguishable faults $\{F_i, F_j \ldots F_k\}$ has been located.
3. No match for E with column vectors in FT is obtained. This result corresponds to the case where the given set of vectors does not allow carrying out fault diagnosis. The set of faults described in the fault table must be incomplete (in other words, the real existing fault is missing in the fault list considered in F.

Example:

	F_1	F_2	F_3	F_4	F_5	F_6	F_7
T_1	0	1	1	0	0	0	0
T_2	1	0	0	1	0	0	0
T_3	1	1	0	1	0	1	0
T_4	0	1	0	0	1	0	0
T_5	0	0	1	0	1	1	0
T_6	0	0	1	0	0	1	1

E_1	E_2	E_3
0	0	1
0	1	0
0	1	0
1	0	1
1	0	1
0	0	0

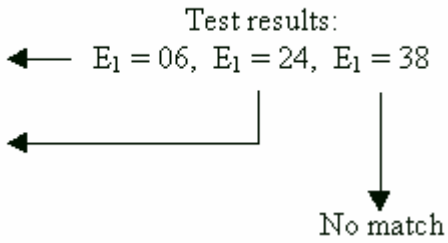


In the example the results of three test experiments E_1, E_2, E_3 are demonstrated. E_1 corresponds to the first case where a single fault is located, E_2 corresponds to the second case where a subset of two indistinguishable faults is located, and E_3 corresponds to the third case where no fault can be located because of the mismatch of E_3 with the column vectors in the fault table.

3.11.2 Fault dictionary

Fault dictionaries (FD) contain the same data as the fault tables with the difference that the data is reorganized. In FD a mapping between the potential results of test experiments and the faults is represented in a more compressed and ordered form. For example, the column bit vectors can be represented by ordered decimal codes (see the example) or by some kind of compressed signature. An example is shown below

No	Bit vectors	Decimal numbers	Faults
1	000001	01	F_7
2	000110	06	F_5
3	001011	11	F_6
4	011000	24	F_1, F_4
5	100011	35	F_3
6	101100	44	F_2



3.11.3 Fault location by structural analysis

Assume a single fault in the circuit. Then a path should exist from the site of the fault to each of the outputs where errors have been detected. Hence the fault site should belong to the intersection of cones of all failing outputs. A simple structural analysis can help to find faults that can explain all the observed errors.

3.11.4. Sequential fault diagnosis methods

In sequential fault diagnosis the process of fault location is carried out step by step, where each step depends on the result of the diagnostic experiment at the previous

step. Such a test experiment is called adaptive testing. Sequential experiments can be carried out either by observing only output responses of the UUT or by pinpointing by a special probe also internal control points of the UUT (guided probing). Sequential diagnosis procedure can be graphically represented as diagnostic tree.

3.11.5. Fault location by edge-pin testing

In fault diagnosis test patterns are applied to the UUT step by step. In each step, only output signals at edge-pins of the UUT are observed and their values are compared to the expected ones. The next test pattern to be applied in adaptive testing depends on the result of the previous step. The diagnostic tree of this process consists of the fault nodes FN (rectangles) and test nodes TN (circles). A FN is labeled by a set of not yet distinguished faults. The starting fault node is labeled by the set of all faults. To each FN k a TN is linked labeled by a test pattern T_k to be applied as the next. Every test pattern distinguishes between the faults it detects and the ones it does not. The task of the test pattern T_k is to divide the faults in FN k into two groups - detected and not detected by T_k faults. Each test node has two outgoing edges corresponding to the results of the experiment of this test pattern. The results are indicated as passed (P) or failed (F). The set of faults shown in a current fault node (rectangle) are equivalent (not distinguished) under the currently applied test set.

Example:

The diagnostic tree [29] in the Figure below corresponds to the example considered in 3.11.2 we can see that most of the faults are uniquely identified; two faults F_1, F4 remain indistinguishable. Not all test patterns used in the fault table are needed. Different faults need for identifying test sequences with different lengths. The shortest test contains two patterns the longest four patterns.

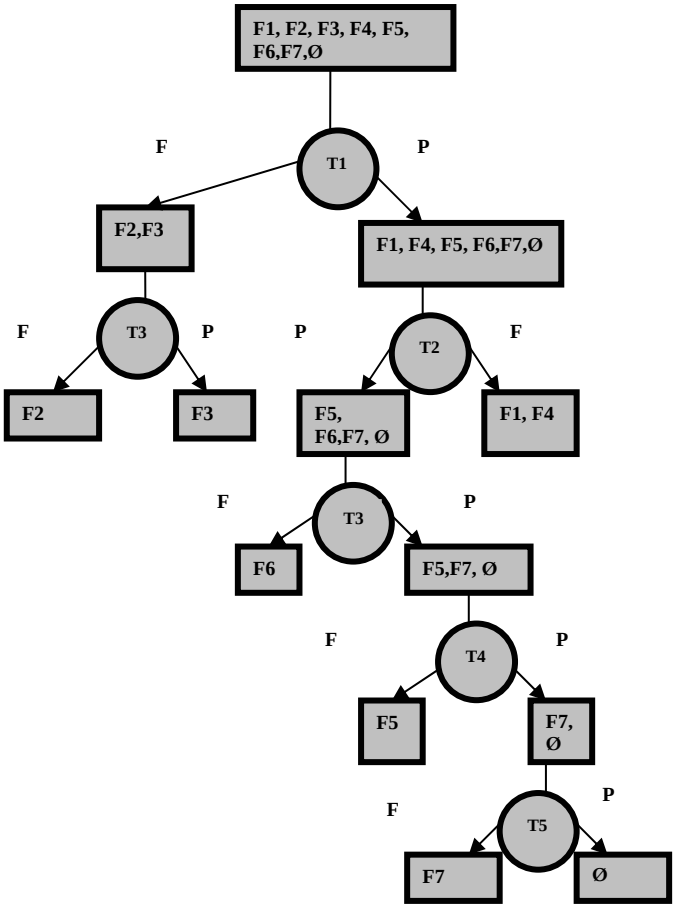


Figure 22: Diagnostic tree

Rather than applying the entire test sequence in a fixed order as in combinational fault diagnosis, adaptive testing determines the next vector to be applied based on the results obtained by the preceding vectors. In our example, if T_1 fails, the possible faults are $\{F_2, F_3\}$. At this point applying T_2 would be wasteful, because T_2 does not distinguish among these faults. The use of adaptive testing may substantially decrease the average number of tests required to locate a fault.

3.11.6. Generating tests to distinguish faults

To improve the fault resolution of a given test set T, it is necessary to generate tests to distinguish among faults equivalent under T.

Consider the problem of generating a test to distinguish between faults F1 and F2. Such a test must detect one of these faults but not the other, or vice versa. The following cases are possible.

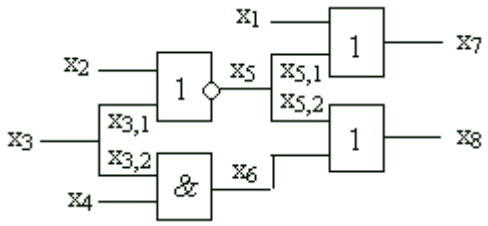
1. F1 and F2 do not influence the same set of outputs. Let OUT (Fk) be the set of outputs influenced by the fault Fk. A test should be generated for F1 using

only the circuit feeding the outputs OUT (F1), or for F2 using only the circuit feeding the outputs OUT (F2).
2. F1 and F2 influence the same set of outputs. A test should be generated for F1 without activating F2, or vice versa, for F2 without activating F1.

Three possibilities can be mentioned to keep a fault F2: $x_k \equiv e$ not activated, where x_k denotes a line in the circuit, and $e \in \{0,1\}$:

1. The value $\neg e$ should be assigned to the line x_k.
2. If this is not possible then the activated path from F2 should be blocked, so that the fault F2 could not propagate and influence the activated path from F1.
3. If the 2nd case is also not possible then the values propagated from the sites F1 and F2 and reaching the same gate G should be opposite on the inputs of G.

Example:

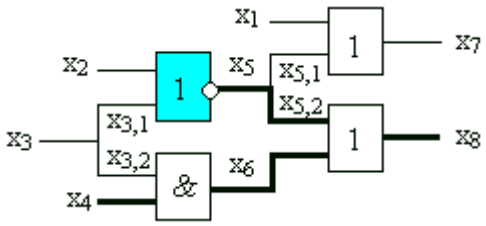


1. There are two faults in the circuit: F1: $x_{3,1} \equiv 0$, and F2: $x_4 \equiv 1$. The fault F1 may influence both outputs; the fault F2 may influence only the output x_8. A test pattern 0010 activates F1 up to the both outputs and F2 only to x_8. If both outputs will be wrong, F1 is present, and if only the output x_8 will be wrong, F2 is present.
2. There are two faults in the circuit: F1: $x_{3,2} \equiv 0$, and F2: $x_{5,2} \equiv 1$. Both of them influence the same output of the circuit. A test pattern 0100 activates the fault F2. The fault F1 is not activated, because the line $x_{3,2}$ has the same value as it would have had if F1 were present.
3. There are the same two faults in the circuit: F1: $x_{3,2} \equiv 0$, and F2: $x_{5,2} \equiv 1$. Both of them influence the same output of the circuit. A test pattern 0110 activates the fault F2. The fault F1 is activated at its site but not propagated through the AND gate, because of the value $x_4 = 0$ at its input.
4. There are two faults in the circuit: F1: $x_{3,1} \equiv 1$, and F2: $x_{3,2} \equiv 1$. A test pattern 1001 consists the value $x_1 \equiv 1$ which creates the condition where both of the faults may influence only the same output x_8. On the other hand, the test pattern 1001 activates both of the faults to the same OR gate (i.e. none of them is blocked). However, the faults produce different values at the inputs of the gate, hence they are distinguished. If the output value on x_8 will be 0, F1 is present. Otherwise, if the output value on x_8 will be 1, either F2 is present or none of the faults F1 and F2 are present.

3.11.7 Guided-Probe Testing

Guided-probe testing extends edge-pin testing process by monitoring internal signals in the UUT via a probe which is moved (usually by an operator) following the guidance provided by the test equipment [31]. The principle of guided-probe testing is to back trace an error from the primary output where it has been observed during edge-pin testing to its physical location in the UUT. Probing is carried out step-by-step. In each step an internal signal is probed and compared to the expected value. The next probing depends on the result of the previous step. A diagnostic tree can be created for the given test pattern to control the process of probing. The tree consists of internal nodes (circles) to mark the internal lines to be probed, and of terminal nodes (rectangles) to show the possible result of diagnosis. The results of probing are indicated as passed (P) or failed (F). Typical faults located are opens and defective components. An open between two points A and B in a connection line is identified by a mismatch between the error observed at B and the correct value measured at A. A faulty device is identified by detecting an error at one of its outputs, while only correct values are measured at its inputs. The most time-consuming part of guided-probe testing is moving the probe. To speed-up the fault location process, we need to reduce the number of probed lines. A lot of methods to minimize the number of probing are available.

Example:



Let have a test pattern 1010 applied to the inputs of the circuit. The diagnostic tree created for this particular test pattern is shown in Figure 23. On the output x_8, instead of the expected value 0, an erroneous signal 1 is detected. By back tracing (indicated by bold arrows in the diagnostic tree) the faulty component NOR- x_5 is located.

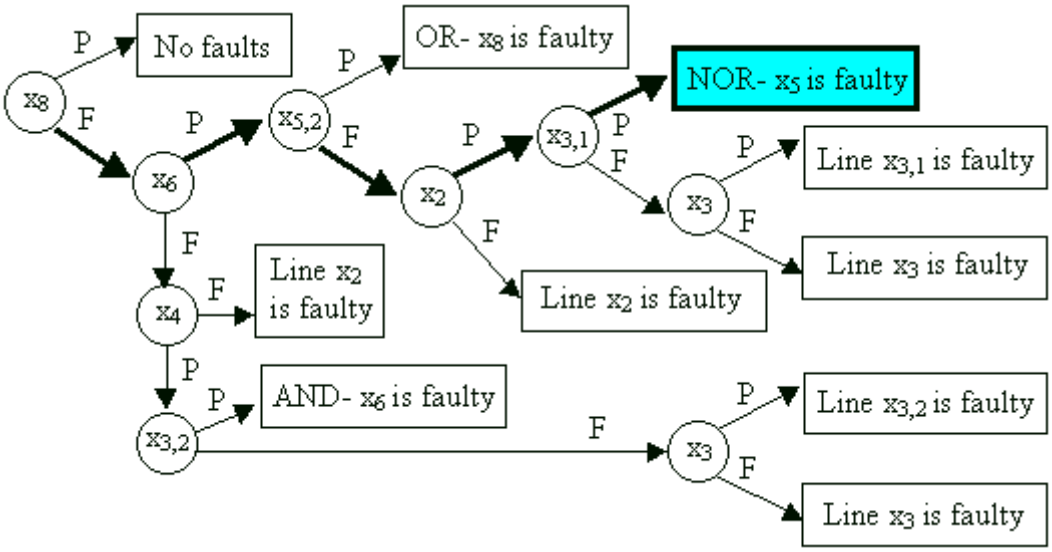


Figure 23: Diagnostic tree

Diagnostic tree allows carrying out optimization of the fault location procedure, for example to generate a procedure with minimum average number of probes.

3.11.8 Fault location by UUT reduction

Initially the UUT is the entire circuit and the process starts when its test fails. While the failing UUT can be partitioned, half of the UUT is disabled and the remaining half is tested. If the test passes, the fault must be in the disabled part, which then becomes the UUT. If the test fails, the tested part becomes the UUT.

3.12 Summary

In this chapter we discuss the general test strategies and diagnostic techniques used in digital circuits. The next chapter deals specifically with Networks on Chips testing.

Chapter 4
Networks on Chips Testing Methods

About this Chapter

It is well know with shrinking geometries, NoCs will be increasingly exposed to permanent and transient sources of error that could degrade manufacturability, signal integrity and system reliability. This chapter deals specifically with NoC testing. The first two subsections deal with general test methods and fault models for NoC. This is followed by description of test data transport, test scheduling, test access mechanism, test interface and test output evaluation. The last subsection describes in brief a test method for NoC routers as proposed by a researcher.

4.1 Test Issues in NoCs

Traditionally, correct fabrication of integrated circuits is verified by post manufacturing testing using different techniques ranging from scan-based techniques to delay and current-based tests. Due to their particular nature, NoCs are exposed to a range of faults that can escape classic test procedures [15]. Such faults include crosstalk, faults in the buffers of the NoC routers, and higher-level faults such as packet misrouting and data scrambling. These faults add to the classic faults that must be tested post fabrication for all integrated circuits (stuck-at, opens, shorts, memory faults, etc.). Consequently, the test time of NoC based systems increases considerably due to these new faults.

Test time is an important component of the test cost and, implicitly, of the total fabrication cost of a chip. For large volume production, the total time that a chip requires for testing must be reduced as much as possible to keep the total cost low. The total test time of an IC is governed by the amount of test data that must be applied and the amount of controllability/observability that the DFT techniques chosen by designers can provide. The test data increases with the complexity of the chip and size, so the option the DfT engineers are left with is to improve the controllability /observability. Traditionally, this is achieved by increasing the number of test inputs/ outputs, but this has the same effect of increasing the total cost of an IC. DfT techniques, such as scan-based tests, improve the controllability and observability of IC internal components by serializing the test I/O data and feeding/extracting it to/ from the IC through a reduced number of test pins. The trade-off is the increase in test time and test frequency, which makes it at-speed test using scan-based techniques difficult. Although scan based solutions are useful, their limitations in the particular case of NoC systems demand the development of new test data generation and transportation mechanism that reduces the total test time and at the same time do not require an increased number of test I/0 pins.

An effective and efficient test procedure is, however, not sufficient to guarantee the correct operation of NoC data transport infrastructures during the lifetime of the IC [15]. Defects may appear later in the life of an IC, due to causes like electro migration, thermal effects, material aging, etc. These effects will become more important with continuous dimension downscaling of the devices beyond 65nm and moving towards the nanoscale domain. The technology projections for the next generations of nanoelectronic devices show that defect rates will be in the order of one to ten percent and defect–tolerant techniques will have to be included in the early stages of design flow of digital systems. Even with defect rates indicated by the International Technology Roadmap for Semiconductors (IRTS) [1] for upcoming CMOS processes, it is clear that correct fabrication is becoming more and more difficult to guarantee. An issue of concern in the case of the communication-intensive platforms such as NoC is the integrity of the communication infrastructure.

4.2 Test methods for NoC Fabrics

The main concern for NoC/SoC test is the design of efficient test access mechanisms (TAM) for delivering the test data to the individual cores under constraints such as test time, test power, and temperature. Among the different TAMs, TestRail [32] was one of the first to address core-based test of SoCs. Recently, a number of different research groups suggested the reuse of the communication infrastructure such as TAM. Vermuelen et al. [33] assumed the NoC fabric as fault-free, and subsequently used it to transport test data to the functional blocks, however for large systems, this assumption can be unrealistic, considering the complexity of the design and communication protocols.

NoCs are built using a structured design approach, where a set of functional cores (processing elements, memory blocks, etc) are interconnected through a data communication infrastructure that consists of switches and links, These core can be organized either as regular or irregular topologies. The test strategies of NoC-based interconnect infrastructures must address three problems: (1) testing of the switch blocks (2) testing of the interswitch wire segments and (3) testing of the functional NoC cores. Test of both routers and links must be integrated in a streamlined fashion. First, the already tested NoC components can be used to transport the test data toward the components under test in recursive manner. Second, the inherent parallelism of NoC structures allows propagating the test data simultaneously to multiple NoC elements under test .Test scheduling algorithms guarantee a minimal test time for arbitrary NoC topologies.

4.3 Fault Models for NoC Infrastructure

When developing a test methodology for NoC fabrics, we need to start from a set of models that can realistically represent the faults specific to the nature of NoC as a data transport mechanism [15]. As stated previously, a NoC infrastructure is built from two basic types of components: switches and interswitch links. For each type of component we must construct test patterns that exercise its characteristics faults**.**

4.3.1 Fault models for NoC interswitch links

One of the proposed fault models for the global interconnects of deep submicron SoCs, accounting for crosstalk effects between a set of aggressor lines and victims lines, is referred to as Maximum Aggressor fault (MAF) [34].

This fault occurs when the signal transition on a single interconnect line (called the victim line) is affected through crosstalk by transitions on all the other interconnect lines (called the aggressors) due to the presence of the crosstalk effect. In this model, all the aggressor lines switch in the same direction simultaneously. The MAF model is an abstract representation of the set of all defects that can lead to any one of following the six crosstalk errors: rising/falling delay, positive/negative glitch and rising/falling speedup.

4.3.2 Fault models for FIFO buffers in NoC Switches

NoC switches generally consist of a combinational block in charge of functions such as arbitration, routing and error control and FIFO memory blocks that serve as communication buffers. Figure 24 shows the generic architecture of a NoC switch. As information arrives at each of the ports, it is stored in FIFO buffers and then routed to the target destination by the routing logic block (RLB).

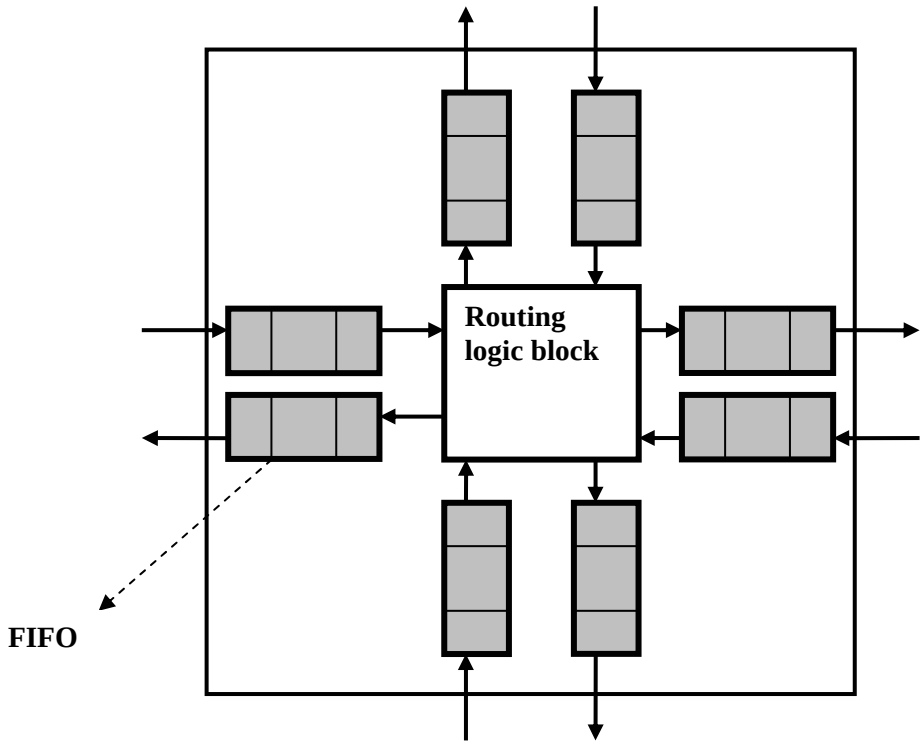


Figure 24: Four port switch generic architecture

The FIFO communication buffers for NoC fabrics can be implemented as registers banks or dedicated SRAM arrays. In both cases, functional test is referable due to reduced time duration, good coverage, and simplicity. The block diagram of a NoC FIFO is show in figure 25. From a test point of view, the NoC–specific FIFOs fall under the category of restricted two-port memories. Due to the unidirectional nature of the NoC communication links, they have one write only port and one read only port. Under these restrictions, the FIFO function can be divided in three ways: the memory cells array, addressing mechanism, and the FIFO-specific functionality**.**

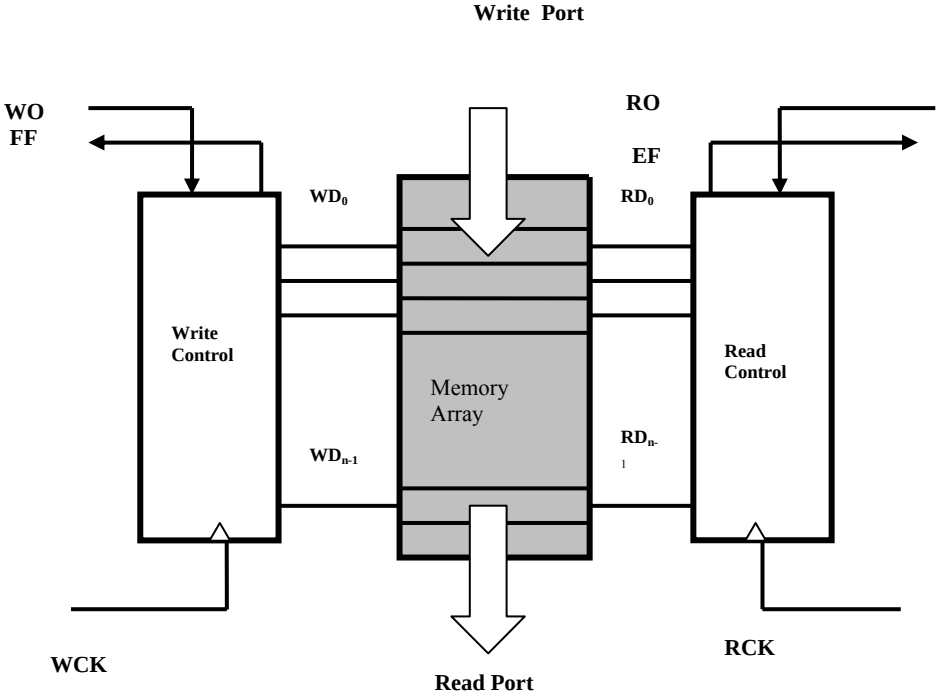


Figure 25: Dual port NoC FIFO

Memory array faults can be stuck-at, transition, data retention, or bridging faults. Addressing faults on the RD/WD lines are also of importance as that may prevent cells from being read / written. In addition, functionality faults on the empty and full flags (EF and FF, respectively) are included in the set of fault models.

4.4 Structural Postmanufacturing Test

Once a set of fault models is selected, test data must be organized and applied to the building modules of the NoC infrastructure. In the classic SoC test, this is accomplished by using dedicated Test Access Mechanisms (TAM) such as Test Rail. Since the NoC infrastructures are designed as specialized data transport mechanism, it is very efficient to reuse them as TAMs for transporting test data to functional cores. The potential advantages when reusing NoC infrastructures as TAMS are the low-cost overhead and reduced test time due to their high degree of parallelism, which allows

testing of multiple cores concurrently. The challenges of testing the NoC infrastructure are its distributed nature and the types of faults that must be considered. A straight forward approach is to consider the NoC fabric as an individual core of the NoC based system, wrap it with an IEEE 1500 test wrapper and then use any of the core based test approaches. More refined methods can be used to exploit the particular characteristics of NoC architectures.

A test delivery mechanism that propagates the test data through the NoC progressively, reusing the previously tested NoC components, was proposed by [35]. The principle is to organize test vectors as data packets and provide, for each router a simple BIST block that identifies the type of packets (test data) and extract/applies the test vectors. Test packets are organized similarly to regular data packets, the difference being a flag in the packet header that identifies the packet as carrying a test sequence. Test specific control information is also embedded into the test packets, followed by the set of test vectors. Figure 26 shows contents of a test packet.

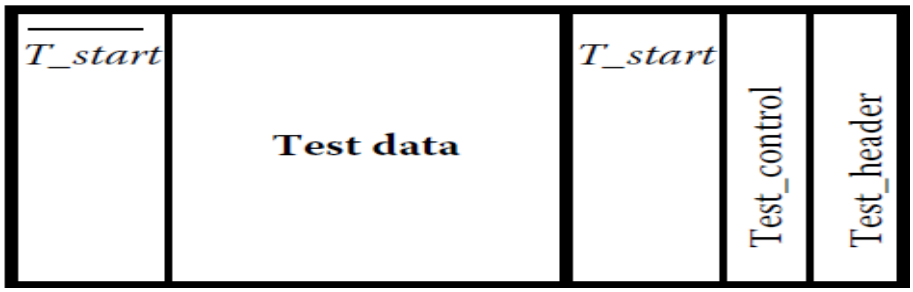


Figure 26: Test packet structure

4.5 Test Data Transport

A system wide test implementation has to satisfy the specific requirements of the NoC fabric and exploit its highly parallel and distributed nature for an efficient realization. In fact, it is advantageous to combine the testing of the NoC interswitch links with that of the other NoC components (i.e., the router blocks) to reduce the total silicon area overhead. However, special hardware maybe required to complement parallel testing features. Each NoC switch is assigned a binary address so that the test packets can be directed to particular switches. In the case of direct connected networks, this address is identical to the address of the IP core connected to the respective switch. In the case of indirect networks not all switches are connected to IP cores, so switches must be assigned specific addresses to be targeted by their corresponding test packets. Considering the degree of concurrency of the packers being transported through the NoC, we can distinguish two cases, described below.
(1) **Unicast Mode**: The packets have a single destination. This is the most common situation and it is representative for the normal operation of an on-chip communication fabric, such as processor cores executing read/write operations from/into memory cores, or micro-engines transferring data in a pipeline. As shown in figure 27, packets arriving at a switch input port are decoded and directed to a unique output port according to the routing information stored in the header of the packet.

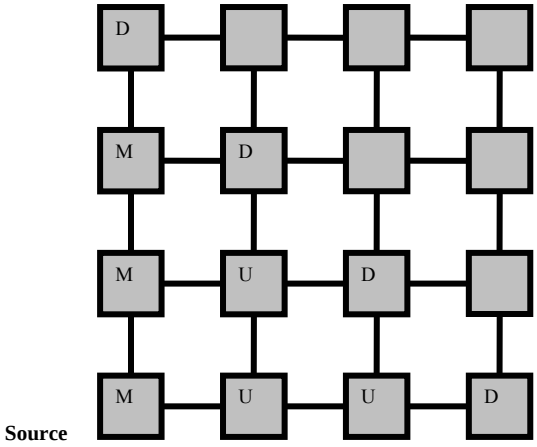


Figure 27: Unicast data transport in a NoC

(2) **Multicast mode**: The packets have multiple destinations. In packets with multicast routing information is decoded at the switch input ports and then replicated identically at the switch outputs indicated by the multicast decoder .Multicast packets can reach their destinations in a more efficient and faster manner than in the case when repeated unicast is employed to send identical data to multiple destinations. Figure 28 shows a multicast transport instance, where the data is injected at the switch source (S), replicated and retransmitted by intermediate switches in both multicast and unicast modes, and received by multiple destination switches (D). The multicast mode is especially used for test data transport purposes, when identical blocks need to be tested as fast as possible. Several NoC platform developed by research groups in industry and academia, feature the multicast capability for functional operation [36, 37]. In these cases, no modification of NoC switches hardware or addressing protocols is required to perform multicast test data transport.

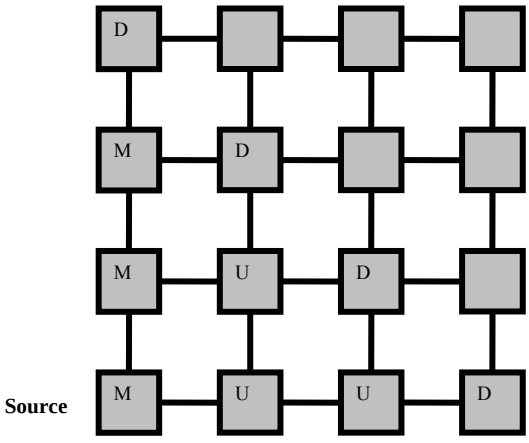


Figure 28: Multicast data transport in a NoC

If the NoC does not possess multicast capability, this can be implemented in a simplified version that only services the test packets and is transparent for the normal

operation mode. As shown in figure 29, the generic NoC structure presented in figure 24 is modified by adding a multicast wrapper unit (MWU) who functionality is explained below. It contains additional demultiplexors and multiplexors relative to the generic switch architecture. The MWU monitors the type of incoming packets and recognizes the packets that carry test data. An additional field in the header of the test packets identifies that they are intended for multicast distribution.

For NoCs supporting multicast for functional data transport, the routing /arbitration logic block (RLB) is responsible for identifying the multicast packets, processing the multicast control information, and directing them to the corresponding output ports of the switch. The multicast routing blocks can be relative complex and hardware-intensive. For multicast test data transport only, the RLB of the switch is completely bypassed by the MWU and does not interfere with the multicast test data flow, as illustrated in figure 29. The hardware implementation of the MWU is greatly simplified by the fact that the test scheduling is done off-line, that is, the path and injection time of each test packet is computed prior to performing the rest operation, Therefore, for each NoC switch, the subset of input and output ports that will be involved in multicast test data transport is known a priori, an the implementation of this feature can be restricted to these specific subsets.

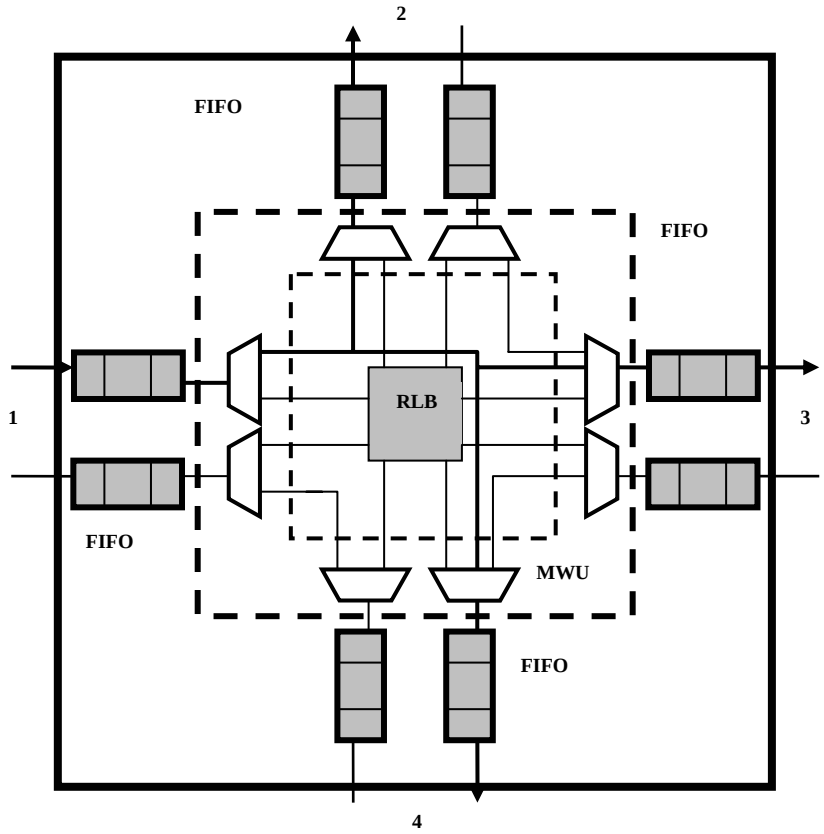


Figure 29: Four port NoC switch with multicast wrapper unit (MWU) for test data transport.

For instance, in the multicast step shown in figure 28, only three switches must possess the multicast feature. By exploring the entire necessary multicast steps to

reach all destinations, we can identify the switches and ports that are involved in the multicast transport, and subsequently implement the MWU for the required switches/ports. The header of a multi destination message must carry the destination node address. To route a multi destination message, a switch must be equipped with a method for determining the output ports towards which the packet must be directed. When designing multicast hardware and protocols with limited purpose such as test data transport, a set of simplifying assumptions can be made to reduce the complexity if the multicast mechanism. This set of assumptions can be summarized as follows:

1. The test data is fully deterministic.
2. Test traffic is scheduled off-line, prior to test application.
3. For each test packet, the multicast route can be determined exactly at all times (i.e., routing of test packets is static).
4. For each switch, the set of I/O ports involved in multicast test data transport is known and ma be a subset of all I/O ports of the switch (i.e.., for each switch ,only a subset of I/O ports may be required to support multicast).

These assumptions help in reducing the hardware complexity of the multicast mechanism by implementing the required hardware only for these switch ports that must support multicast. For instance in the example of figure 29, if the multicast feature must be implemented exclusively from input port 1 to the output ports 2,3,4, then only one demultiplexor and three multiplexors are required.

Since the test data is fully deterministic and scheduled off-line, the test packets can be ordered such that the situation where two (or more) incoming packets compete for the same output port of the a switch can be avoided .There fore, no arbitration mechanism is required for multicast test packets, Also, by using this simple addressing mode, no routing tables or complex hardware is required.

The lack of I/O arbitration for the multicast test data has a positive impact on the transport latency of the packets. A test-only multicast implementation has lower transport latency than the functional multicast, because the only task performed by the MWU block is routing. The direct benefit is a reduced test time compared to the use of fully functional multicast, proportional to the number of processes that are eliminated. The advantages of using the simplified multicasting scheme are reduced complexity, lower silicon area required by MWU, and shorter transport latency for the test data packets.

4.6 Test Scheduling

Test scheduling is one of the major problems in the embedded core testing. A general from S of this problem can be formulated as follows: for an NoC-based system N, a maximal number P of input/output ports, a set T of test sets (each core may have multiple test sets, either deterministic or random, or functional test for the processor cores), a set C of constraints, and a set R of test resources (dedicated TAM, BIST engines, etc.) determine a selection of input/output ports, an assignment of test resources to cores , and a schedule of test sets such that the optimization objectives

are minimized and all the constraints are met. Note the objective can be any cost factor such as test time and hardware overhead. A basic subset of this problem S is problem $S_{0\ where}$ only core tests are optimized under some constraints. it was proved as N P complete in[38]. S and other subsets of S can also be similarly proved as NP complete.

In a NoC based system, test scheduling can be done in a manner of either preemptive or nonpreeemptive. As in general SOC, an optimized schedule should maximize the test data parallelism. In an NOC –base system, this is done through the exploration of network parallelism so that all available communication resources (channels, routers and interfaces) are used in parallel to transmit test data. In preemptive test scheduling, test data are transformed in test packets, which are transmitted through the network in a such a way that the test packet contains one test vector or one test response, Since each test vector or test response can be scheduled individually, the network parallelism has privilege over the core test pipeline, and the test of the core can be interrupted. As a result, the pipeline of the core's scan-in and scan-out operations cannot be maintained.

Preemptive testing is not always desirable in practice especially for BIST and sequential circuit testing. In addition, it is always desirable that the test pipeline of a core is not interrupted – that is the n^{th} test vector will be shifted into the scan chains as the $(n-1)^{th}$ test response is shifted out, such that the test time is minimized. However, in the case of preemption, the test pipeline has to be halted if either the test vector or test response packet cannot be scheduled because of the unavailability of test resources i.e. channels and input/output ports. This will not only increase the complexity of wrapper control but also cause potential increase in test time.

A nonpreeemptive schedule maintains the test pipeline so that the wrapper can remain unchanged and the test time can be potentially reduced. In this approach, the scheduler will assign each core a routing path, including an input port, an output port, and the corresponding channels that transport test vectors from the input to the core and test responses from the core to the output in the form of packets. Once the core is scheduled on this path, all the resources (input/output, channels) on the path will be reserved for the test of this core until the entire test is completed. Test vectors will be routed to the core and test responses to the output in a pipelined fashion. Therefore, in this nonpreeemptive schedule, the test core is identical to a normal test and the flow control becomes similar to circuit switching. It has been shown that usually nonpreeemptive scheduling can yield shorter test time compared to preemptive scheduling [38]. This is because the test pipeline can be maintained, scan-in and scan-out can be overlapped, and hence, test time is reduced, it can also avoid the possibility of resource conflict. The complexity of nonpreeemptive scheduling algorithm is much lower than of a preemptive scheduling algorithm, because the minimum manageable unit in scheduling is a set of test packets n the former, instead of a single packet in the latter.
In practice, it is more realistic to have both preemptive and nonpreeemptive test configurations in testing as system. It is also necessary to consider various constraints such as power, precedence, and multiple test sets such as external test and BIST.

A preemptive schedule can be useful can be useful under these requirements. For instance, excessive power dissipation and inadequate heat convection/conduction can cause some cores to be significantly hotter than other, so called hot spots. Applying the entire test suite continuously can lead to dangerous temperature on these cores. In this case, the test suite can be split into several test sessions (or even single test vector in extreme cases) that can be scheduled individually in a preemptive manner. Sufficient time can be allowed between test sessions for hot spots to be cooled down.

4.7 Test Access Methods and Test Interface

Test access and test interface design in an NOC-based system need techniques different from those in conventional SoCs. A typical instance is the multiprocessor system discussed in [39]. Each node is such an NOC-based system can contain a processor core and the corresponding router or switch buffers, etc. Testing of a processor –based system usually mandates both deterministic test and functional test. Most functional test approaches such as software based BIST [40] can be applied. However all test patterns and test responses must be organized into sets of packets and all necessary network interfaces need to be added. For deterministic tests, nodes can be organized into groups, and each group can be tested using a conventional boundary scan approach. Figures 30 and 31 shows the test configuration in this scheme when nodes are organized into 1*1 and 2*2 groups, respectively. Nodes in the same group share the test infrastructure for boundary scan (test access port [TAP] controller, I/O cells, etc.) and are tested in parallel. Since the nodes are identical, each bit of test data is broadcasted to all nodes in the group. Test responses from all nodes in the group can be processed on-chip by feeding to a comparator as shown in figure 31.

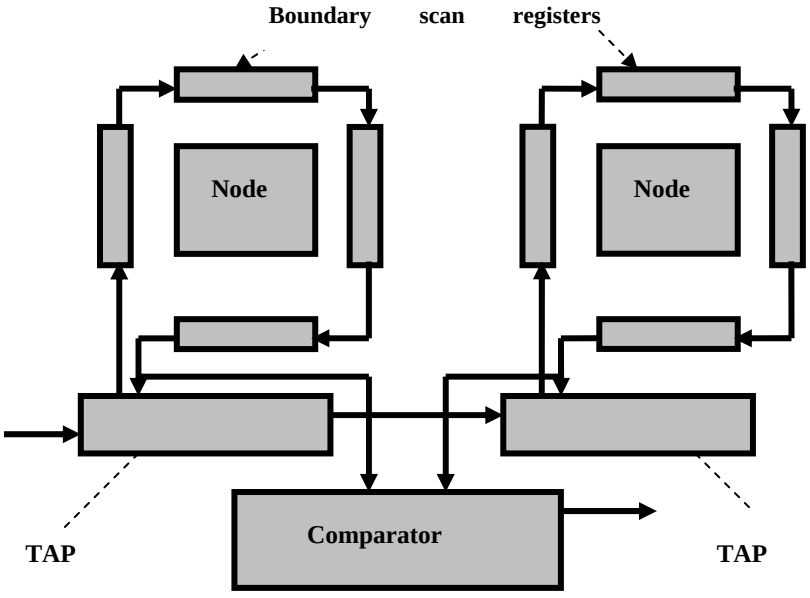


Figure 30: Testing identical nodes in NoC using boundary scan in group of 1*1

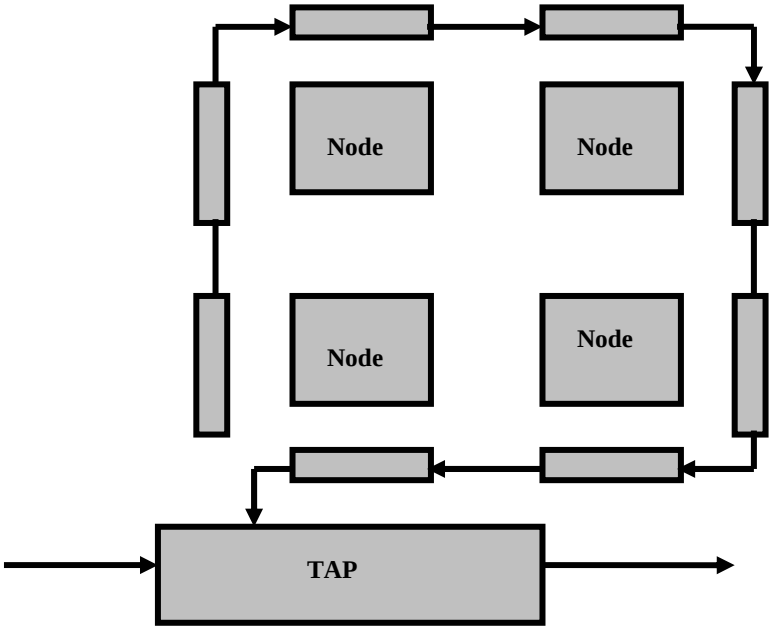


Figure 31: Testing identical nodes in NoC using boundary scan in group of 2*2

To reuse the network to transport test data, a test interface has to be established to handle both functional protocol from network and test application to the core. A wrapper is therefore needed for each core as an interface. Since the core and network may use different protocols, the wrapper must be extended to incorporate the standard IEEE 1500 test modes. This includes modification on the test wires, wrapper cells, and test control logic. The TAM port in the 1500 wrapper should be replaced by a port connecting to the network. The control logic should include the process of network protocols. Further, wrapper cell should be correspondingly modified to implement the protocol. An instance of such a wrapper cell is shown in figure 32, as well as a standard wrapper cell is shown in figure 33 [41].Note that both the cells need functional/scan input/output terminals and a few control terminals for the Muxes. When compared with the traditional 1500 wrapper cell, the modified cell has an additional mux, Terminal prot_in receive the required values fro protocol operation from the control logic. In test mode t, the terminal prot_mode is asserted to 1 to ensure that test signal does not interfere with the functional protocol; the actual protocol is implemented in the control logic. Other modes in the 1500 wrapper can be maintained.

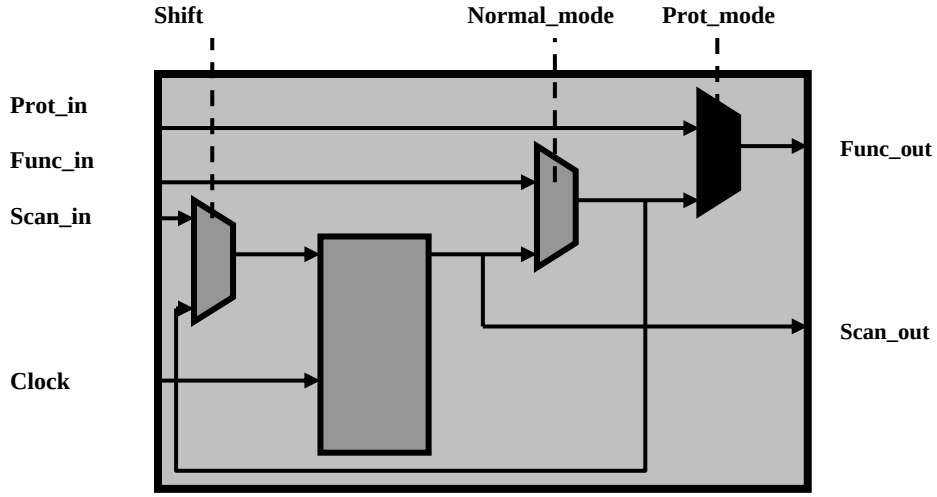


Figure 32: Modified wrapper cell for NoC based system

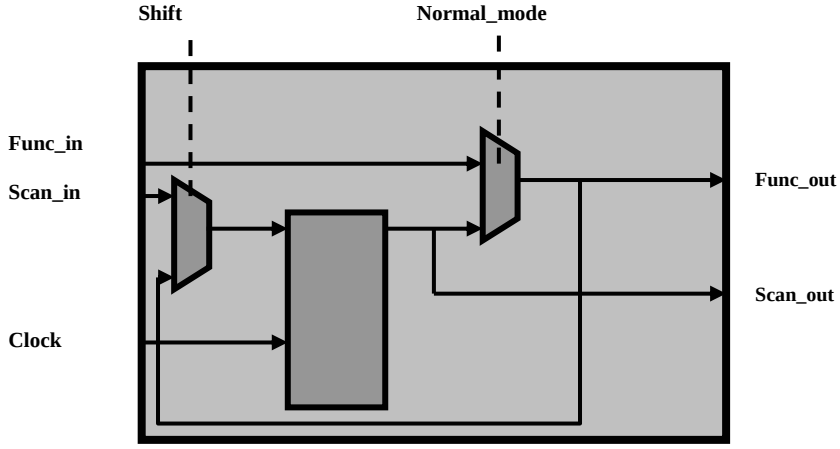


Figure 33: Standard 1500 wrapper cell

4.8 Test Output Evaluation

In classical core-based testing, test data is injected from a test data is injected from a test source, transported and applied to the core under test, and then the test output is extracted and transported to the test sink for comparison with the expected response. A more effective solution was first proposed by Grecu et al. [35], where the expected data is sent together with the input test data, and the comparison is performed locally at each component under test. A clear advantage of this approach is that, because there is no need to extract the test output and to transport it to a test sink, the total test time on the NoC infrastructure can be significantly reduced. Moreover, the test protocol is also simplified, because this approach eliminates the need for a flow control of test output data (in terms of routing and addressing). The trade-off is a small increase in hardware overhead due to additional control and comparison circuitry, and increased size of the test packets (which now contain the expected output of each test vector, interleaved with test input data).The test packets are

processed by test controller (TC) blocks that direct their content towards the I/Os of the component under test (CUT) and perform the synchronization of test output and expected data. This data is compared individually for each output pin and, in case of a mismatch; the component is marked as faulty by raising the pass/fail flag. The value of this flag is subsequently stored in pass/fail flip-flop, which is a part of a shift register that connects pass/fail flops of all switches. The content of this register is serially dumped off-chip at the end of the test procedure.

4.9 Functional Test of NoCs

The architectural and technological complexity of NoC communication infrastructure demands the application of higher level tests for increasing the level of confidence in their correct functionality. High-level fault models for NoC infrastructures are developed taking into account the operation of NoC components (routers) and the services that a NoC must provide in terms of data delivery. From a functional point of view, the services that a NoC provides can be categorized as follows:

- Routing services. These include forwarding data from an input port of a router to an output port according to the routing information embedded in the data packets
- Guaranteed /best effort services. Data must not only be routed correctly but also performance requirements specified according to the QoS parameters in terms of throughput/latency must be satisfied for a NoC to operate correctly.
- Network interfacing. A NoC must be able to inject/eject data correctly at its interfaces (NIs, network interfaces), where the functional cores (processing elements, memory blocks, and other functional units) is plugged into the NoC platform. NIs also has a role in providing QoS guarantees by reserving a route from a source NI to a destination NI in case of GT (guaranteed throughput) data.

Three high-level fault types can be defined according to the functionality described above:

1. Routing faults: Their effect is misrouting of data from an input port to an output port of the same router.
2. Router QoS faults: Their effects consist of QoS violations by a router or set of routers.
3. NI faults: Model faults at interaction between the functional cores and the NoC data transport fabric. These faults can be packetization/depacketization faults or QoS faults.

NoC test based on functional fault models has several advantages compared to structural test, the most important ones being lower hardware overhead and shorter test time mainly due to a reduced set of test data that has to be applied. For a satisfactory fault coverage and yield, both structural and functional tests are required for NoC platforms.

4.10 Testing of Routers

Routers are used to implement the functions of flow control, routing, switching, and buffering of packets for an on-chip network. Figure 34 shows a typical structure of a NoC–based design [41]. As shown in the figure 34 router testing can be considered the same away as testing a sequential circuit. However, special property of an on chip network is its regularity. In [41], an efficient test method has been developed based on partial scan and IEEE 1500-compliant test wrapper by taking advantage of the NoC regularity. In [41], router testing has been dealt with in three parts; the testing of each router, the testing of all routers (without considering network interfaces and interconnects), and the testing of wrapper design. Testing of a router consists of testing the control logic (routing arbitration, and flow control modules) and input FIFOs [42].
Control logic testing can be done using traditional circuit testing methods such as scan, A smart way to test each FIFO is to configure the first register of the FIFO as part of a scan chain, and other registers can be tested through this scan chain. Since FIFOs are generally not deep, this method proves to be very efficient. Since routers are identical, all of them can be tested in parallel by test pattern broadcasting as shown in figure 35, Comparator implemented by XOR gates can be used to evaluate the output response. The comparator logic also supports diagnosis.

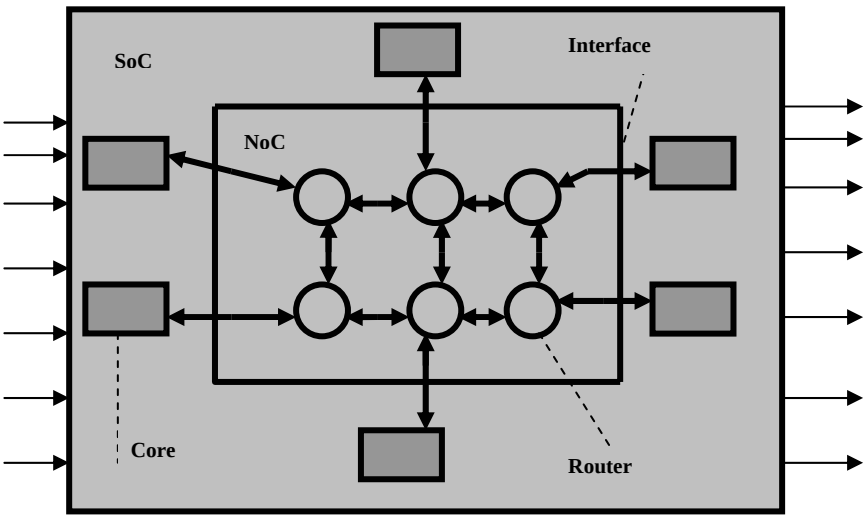


Figure 34: A NoC based system

To support the proposed test strategy, an IEEE-1500 compliant test wrapper is designed to support test pattern broadcasting and test response comparison as shown in figure 36. For example, all SC1 scan chains of these routers share the same set of test patterns. Similarly, all Din [0] (i.e., Din-R0 [0],… Din-Rn [0] data inputs of these routers share the same set of test patterns. As figure 36 shows these wrappers also supports test response comparison for scan chains and data outputs. Finally, the diagnosis control block can activate diagnosis. Simulation results demonstrate that the proposed router test method achieves the goals of small hardware overhead (about

8.5% relative to router hardware), small number of test patterns (several hundreds) because of test pattern broadcasting, and small amount of test application time (several tens of thousands of test cycles) using multiple balanced scan chains and test pattern broadcasting.

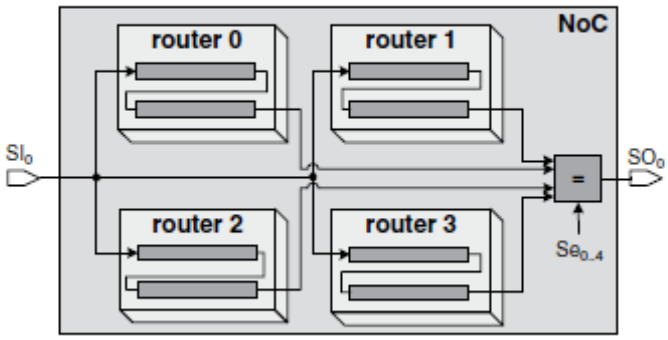


Figure 35: Testing multiple identical routers

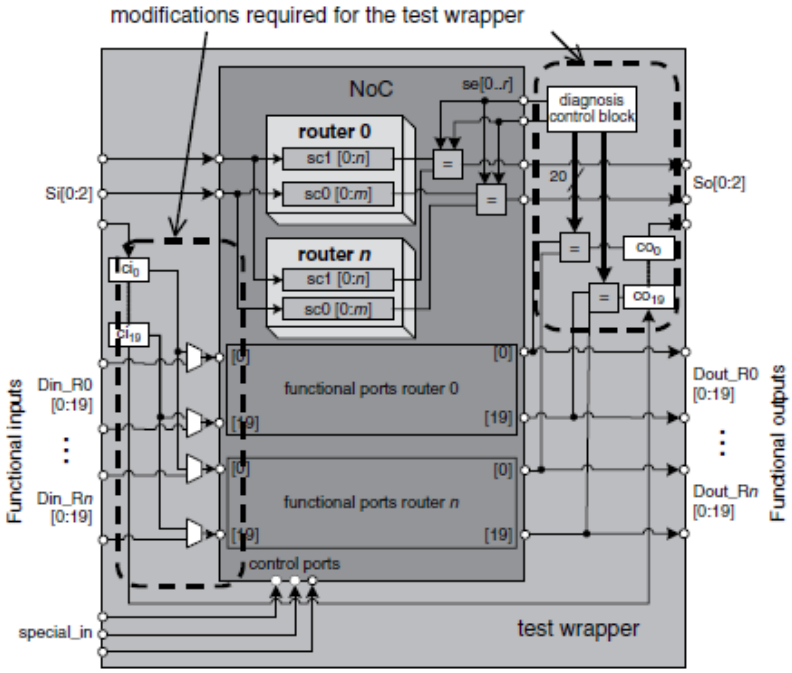


Figure 36: Test wrapper design

4.11 Related Work

In [7] the authors suggest a NoC test strategy which accomplishes a test time that is more or less constant independent of the number of nodes in the 2D-mesh NoC equipped with switches of type Nostrum [44]. The overall BIST strategy is to use the NIs for test of the NoC and also for test of the resources. They are used as a test wrapper around the NoC and as an EATE (Embedded Automated Test

Equipment).With this approach the Resources and the NoC are totally isolated from each other during testing.

All the BIST logic needed to test the NoC is embedded in the NIs. The NIs can be reused as individual eATEs also when testing the resources, if the BIST logic need for that is also placed in the NIs. Each eATE applies test stimuli and collects test results. The collected test results are analyzed after the test has been carried out.

During test of the NoC the switches are used in two different modes. A switch is either the device under test (DUT) or a test wrapper during test of the neighboring switches. With this approach the test of the NoC is divided into two similar test steps. For a 3*3 NoC this means that in the first step: Five switches are in DUT mode and the remaining four switches are in a test wrapper mode .In the second step, the five switches that earlier were in DUT mode now has changed been changed to be in test wrapper mode and the four switches that earlier were in test wrapper mode now is changed to DUT mode. The switches that are used as test wrappers also can carry out some internal tests of the switch at the same time as being used as a test wrapper.

The only test logic inserted into each switch is a number of 2-to-1 multiplexor. They are inserted to be able to divide the total test of the DUT into a number of smaller stand-alone tests. Double layer of multiplexors are inserted to either enable to carry out different sub-test concurrently or to assure that all parts of the NoC are executed in functional mode somewhere during the test.

One of the main drawbacks of this method may be that the test time is prolonged when applied to switches that the one used by Nostrum, since they probably are more complex designs.

4.12 Summary

NoC has become a promising design paradigm for future SoC design, and cost efficient test methodologies are imminently needed. Research on NoC test is still premature when compared to industrial needs because of the limited support for various network topologies, routing strategies, and other factors. Future research and development are need to provide an integrated platform and set methodologies that are suitable for various networks such that the design and test cost of the overall NoC based system (both cores and network) can be reduced. The next chapter deals with a NoC router test methodology proposed by the author, which achieves high fault coverage.

Chapter 5
External Test Approach for NoC Switches

About this Chapter

This chapter describes in details the testing, DfT and diagnosis approach adopted by us. The first section of this chapter describes the motivation for the work while the subsequent section elaborates the synthesized switch architecture. The next sections deal with introducing the concept of fault model and gets into the details of our external test pattern application method. The subsequent section elaborates the Design for testability (DfT) structures used to inject the test patterns. The final section describes the diagnosis of faulty links in the switch.

5.1 Motivation

Network-on-a-Chip (NoC) has emerged as a new design paradigm to replace traditional bus architectures of microelectronic chips. The main motivator for adopting the NoC approach in designing System-on-a-Chips lies in high bandwidth, low latency and a scalable communication infrastructure. Due to the regularity of NoC architectures and the high degree of parallelism in communication it is intuitive that external test configurations, diagnosis and graceful degradation methods should be applicable. The goal of the thesis is to propose a method for targeting manufacturing faults in the switches (i.e. routers) of NoCs based on test configurations to be applied from the boundaries of the network.

Furthermore, the parallel infrastructure introduces a sort of routing redundancy to the system in the sense that broken connections can be avoided and packets can be routed via alternative paths. Thus, a very natural scheme for graceful degradation in NoCs would be to reconfigure the routers in the network by switching off the faulty links. This can be done either after the post-manufacturing test, on the field or on-line, during the normal operation of the chip. This particular aspect will be considered in the thesis by proposing diagnosis of faulty links in the NoC routing network.

The thesis targets regular two-dimensional mesh-like NoCs. As will be shown in this study it is not possible to achieve high fault coverage without supporting test application by dedicated Design-for-Testability (DfT) circuitry. Furthermore, it will be shown that the fault coverage in switches in such networks is highly dependent on the exact logic implementation of its building blocks: crossbar and the input/output buffers.

Testing of NoCs is a new challenge to be overcome by the research community. Over the years, a number of NoC test approaches have been proposed. Vast majority of them are based on implementing design-for-testability structures (i.e. wrappers, scan-paths, built-in test pattern generators and compactors).

In [4], Aktouf proposes the use of boundary scan wrapper for NoC testing routers. However, Amory et al. [41] point out that the use of standard DfT solutions for networks-on-a-chip results in a prohibitively large area overhead. The obvious drawbacks of all of the above methods are the required silicon area for test structures and lack of support for at-speed and functional system testing. In [7], Petersen et al. introduce an idea of near-constant-time testable (C-testable) built-in self-test based test configuration. However, this idea has never been implemented, the required overhead area is prohibitively large and there is no reference of the test quality achieved.

What has been missing previously is a scalable external test approach relying mainly on the NoC network's own high-throughput infrastructure for test access. Test configurations to handle switch matrixes of reconfigurable hardware devices have been developed (e.g. [8]). However, despite apparent similarities, the ideas cannot be implemented for packet-switched routers. The main difficulty is that test configurations in NoCs are dependent on routing algorithms and thus it is not possible to activate arbitrary test paths similar to configuring FPGA switches.

In the field of diagnosing NoC faults, there have been several works that take advantage of on-line checkers. Nurmi et al. [51] developed a fault-diagnosis-and-repair (FDAR) system built in to the network. The method is based on system-level fault models accompanied with parity checking. However, its implementation requires 20 % of area overhead. Grecu, Ivanov et al. [52] propose the use of code-disjoint switches for error detection in NoC communication fabrics. Hosseinabady et al. [53] propose a concurrent testing method for NoC switches requiring on-chip signature analysis, a dedicated test-bus to reach test vectors and collect their responses, and wrappers for test application. A different approach has been introduced by Bhojvani and Mahapatra who propose integration of test infrastructure IPs into the network in order to provide on-line test functionality [54].

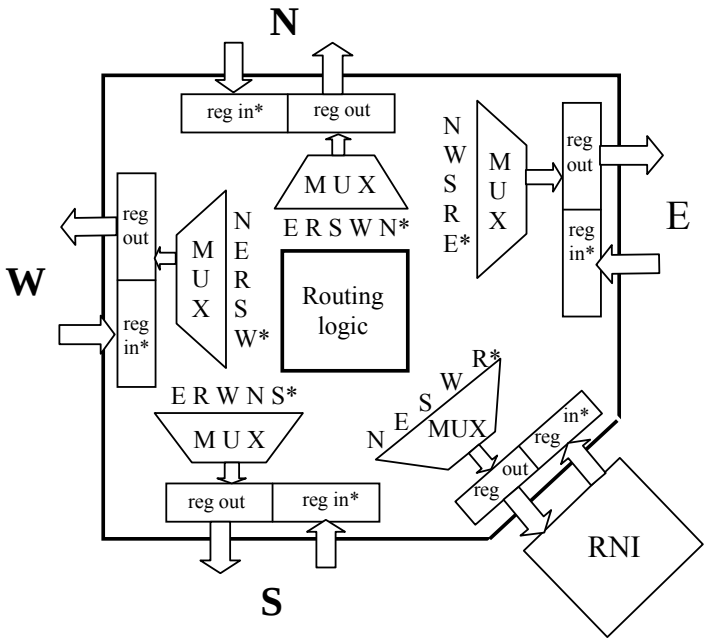
The main shortcoming of the above-mentioned approaches is that they are either based on Design-for-Testability (DfT) structures requiring large area overhead or they rely on traditional scan-based methods. Due to the fact that modern SoCs contain several clock domains, it might be difficult or even impossible to test them at full operation speed with scan-based test approach. Functional-level testing usually improves the quality of testing because it allows testing with true test cases at full operation speed [55].

5.2 Targeted NoC Architecture

We can view the general case of a NoC as a set of cores (*resources*) communicating over a packet-switched network. The network consists of routers (*switches*) connected by interconnect lines. Resources have access to the network via a dedicated bus interface called the Resource Network Interface (*RNI*). A broad variety of NoC switch

architectures has been proposed in the past (e.g. [17, 33, 43, 44]) and there exist different network topologies, switch architectures and routing algorithms.

The switch can connect to five directions: N (north), E (east), R (resource), S (south) and W (west). We consider a concept known as *deflective routing*, where the packet traffic does not have to be queued into input/output FIFOs. In such networks, packets may arrive at the destination in a different order than they were sent and they are subsequently stored and reordered at the RNI. The flits of transmitted packets are buffered using output registers. Input buffering is optional. Similar concept has been implemented e.g. in NOSTRUM [44] and INTEL [17] switches.
Deflecting switches can be categorized into *bouncing* and *non-bouncing* ones depending of whether feedback routing is allowed. Routing decisions are performed locally at the switch and are guided by the *routing logic*, which can be of arbitrary complexity. The deflecting switch architecture described above is presented in figure 37.



* - feedback routing and input registers optional

Figure 37: The general architecture of the switch

The architecture of the control box is given in Figure 38. The control box is a purely combinational circuit. It consists of three subsequent stages. First, the addresses of packages are decoded in five address decoders (one for each input direction plus the resource). Then, priority sorter resolves any possible conflicts calculates the actual output directions of the packages to be sent and also finds Buffer_ready and Write_request values to be signaled to the neighboring switches. Finally, control signals (register enables, mux selects) are decoded from the direction codes.
The following notation is used in Fig. 38:

*XI – X address In
*YI – Y address In
*WI – Write In

*RI – Read In
*XO – X address Out
*YO – Y address Out
*DIR – Output Direction
*BR – Buffer Ready
*WR – Write Request
*OF – Output Final direction

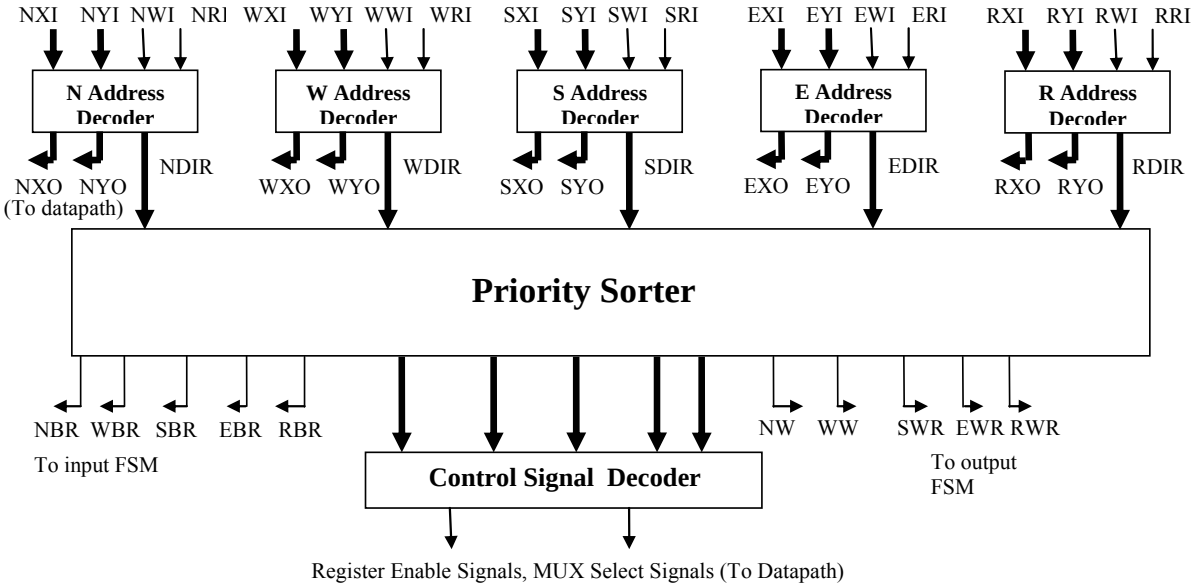


Figure 38: Architecture of the switch control box

Figure 39 presents the data transmission through the NoC switch. Data is fed into an input register and based on its address fields forwarded to corresponding mux and output register. The data transmission is controlled by the *control box*, which generates the datapath control signals (i.e. register enables (Enable) and mux selects (Sel)). In addition, the control box handles the conflicts that occur when data from different inputs are to be sent to the same output direction. Handshaking between neighboring switches is organized via *Ready* and *Write* signals. The control box generates *Buffer_ready* and *Write_request* signals, which are fed to D-flip-flops to provide the *Ready_out* and *Write_out* signals, respectively. These signals are transmitted to the neighboring switches.

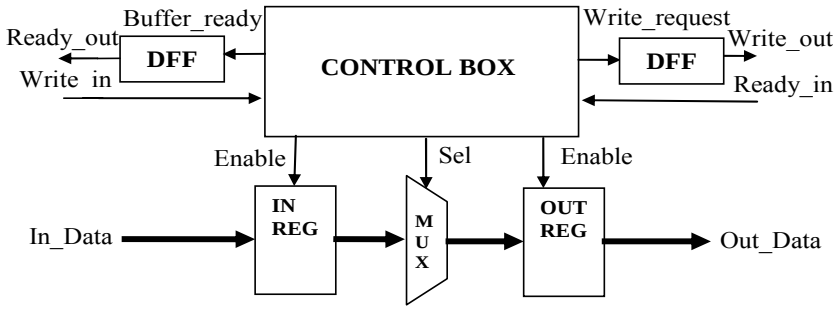


Figure 39: Data transmission scheme for the NoC switch

For the purpose of evaluating the functional diagnosis approach proposed, a generic parametrizeable VHDL description of the deflecting NoC switch has been created. It is possible to set data width, and the number of address bits in the address fields. (In our experiments we selected 128-bit wide data buses). In addition, the following architectural options could be selected: bouncing and non-bouncing switches, binary and one-hot multiplexers and buffering of input data to registers. Total eight types of switches were synthesized. A family of 8 switches representing different possible architecture configurations was synthesized as shown in the table 1 below.

Table 1: Eight variants of switch implementations

	with input registers	w/o input registers
and-or MUX	bouncing / non-bouncing	bouncing / non-bouncing
one-hot MUX	bouncing / non-bouncing	bouncing / non-bouncing

Table 2: Area requirement for a 128-bit switch in equivalent gate counts

Circuit	Combinational Area	Non-combinational Area	Total Area
5 Ready_out DFFs	**0**	**5x9=45**	**45**
5 Write_out DFFs	**0**	**5x9=45**	**45**
5 Address decoders	**5x104=420**	**0**	**420**
Priority sorter	**213**	**0**	**213**
Contr. signal decoder	**150**	**0**	**150**
Switch datapath	**8435**	**11520**	**19955**
Total area:	**9318**	**11610**	**20928**

The total size of the synthesized NoC switch in logic gate equivalents is about 21 k gates. More than 95 % of the total area is consumed by the switch datapath, i.e. input/output registers and muxes. The following Section presents a functional model and an external test approach targeting the datapath parts of deflective switches embedded into a mesh- like network. The area requirement for a 128 bit switch is shown in the table 2.

5.3 Functional Fault Model for Crossbar Switches

In this Section we will explain the functional fault model that is applied to multiplexers and registers in the switch datapath. Similar concept has been successfully implemented in register-transfer level automated test pattern generation (e.g. [45]). Note that minimal tests for different stand-alone MUX implementations were given by Makar and McCluskey in [46]. However, the situation in NoCs is different due to the fact that switches are embedded into the network. Thus, dedicated

test configurations have to be implemented similar e.g. to FPGA interconnect testing approaches [47]. Differently from the latter in the external test approach presented in this paper the test configurations are based on the routing algorithm implemented in the switches. For the current case it is the deterministic, or XY routing.

The method presented in the paper is based on a functional fault model aimed at covering the structural faults in the datapath of the switch. The datapath is essentially a crossbar switch. For the crossbar multiplexers, an approach shown in Figure 40 is used, where the value at selected input is distinguished from the values at other inputs of the MUX (in current method T_1 is the inverted value of T_2). In order to fully cover the structural faults in the multiplexer, tests for each address value have to be performed. An additional constraint is that all bit values in the selected data input must be covered.

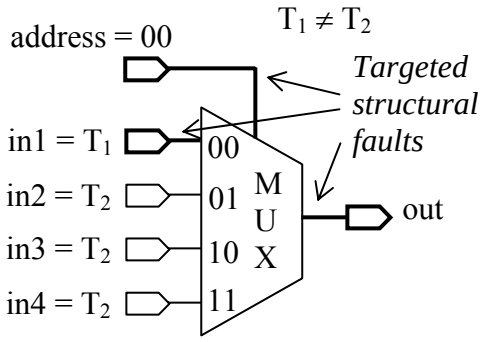


Figure 40: Functional fault model for muxes

The functional fault model used in current paper is based on the following:

1. A functional approach is used to target the multiplexers in the switch, where the value at the selected mux input is distinguished from the values at other inputs. For example, if the test sequence given to the selected mux input is T_1 then sequence T_2 has to be applied to the other inputs of the mux.

2. Different test configurations are applied to cover all the switching modes (i.e. all the address select values) of the muxes. Pipelining of test sequences transmitted to subsequent switch stages helps to keep the overall test length minimal.

5.4 Test Configurations for Mesh-Like NoCs

In this Section we introduce an external test approach implementing the required test configurations in order to apply the functional fault model presented in previous section to the switches embedded into a two-dimensional mesh network. By external test we consider the case where test patterns are applied at the border I/Os of the network, as opposed to using scan paths and wrappers for test access. We will show that testing of switches in the network has a complexity, which grows only linearly

with the number of switches in the NoC. Thus, external testing of packet-switched networks is a viable alternative to the popular design-for-testability based approaches. The experiments presented in the next chapter prove that, in fact, higher fault coverage with shorter test application times will be achieved by functional testing.

The paper proposes a scalable test method, which pose the following two requirements for the network under test:
1. Network topology must be a two-dimensional mesh.
2. The routing algorithm must have support for deterministic XY routing.

In current work it is assumed that the switch includes support for deterministic XY routing. Other, more complex routing schemes, may be additionally implemented alongside the XY. Alternatively, if XY is not among the available routing schemes of the switch then a design-for-testability structure similar to the one forcing YX routing presented in this paper may be introduced . The additional overhead area incurred is 0.4 % of the entire circuit.

5.5 Preliminaries

As it was mentioned before, the external testing method presented in the paper is based on a functional fault model aimed at covering the structural faults in the multiplexers and registers of the switch. Test data transmitted to the switch input buses is composed of *checkerboard patterns* and their complements (i.e. bit vectors 01010...1 and 10101...0, respectively).

The network under test is an n×n mesh of switches. Let the interconnect bus between switches be k bits wide. We define the following test vector values for the NoC busses:

$x = \{01\}^{k/2}$, i.e a checkerboard pattern of interleaved 0s and 1s;

$\neg x = \{10\}^{k/2}$, a complement of x;

$e = \{1\}^{k}$, a vector with all-ones;

$z = \{0\}^{k}$, a vector with all-zeros;

$u = \{-\}^{k}$, a vector with don't-care values.

Let X be a test sequence consisting of three test vectors X=(x, ¬x, x) and ¬X be the complement of X (i.e. ¬X=(¬x, x, ¬x)).

The main rationale behind the test method proposed in this paper is based on the following:

1. A functional approach is used to target the multiplexers in the switch, where the value at the selected mux input is distinguished from the values at other inputs. For example, if the test sequence given to the selected mux input is X then sequence ¬X has to be applied to the other inputs of the mux.

2. Different test configurations are applied to cover all the switching modes (i.e. all the address select values) of the muxes. *Pipelining of test sequences* transmitted to subsequent switch stages helps to keep the overall test length minimal.

5.6 Targeted Fault Models

The functional test approach proposed in this paper is targeting the logic-level stuck-at fault coverage. However, it allows coverage for other important fault models. Note, that since the topological layout of the routing wires between NoC switches is usually regular and may be backannotated from the physical implementation data, the Wagner's logarithmic algorithm [48] is not needed to cover all the opens and shorts in adjacent interconnect lines. We don't have to consider the general case of wiring because in NoCs we are dealing with regular layout structures, where interconnect lines connecting two switches are normally laid out as straight wires ordered in the same way as bit positions of the logical bus. In fact, a checkerboard pattern followed by its complement will be enough to guarantee testing of all possible opens and shorts between adjacent wires connecting the switches. An additional checkerboard pattern (i.e. test sequence the test sequence X or ¬X) would provide for, both, rising and falling edge transitions at each bit position and, thus, already for the full coverage of delay faults at the inter-switch wires.

Let us denote the length of the interconnect test sequence for one path by t. For test sequence X =(x, ¬x, x) the test length t=3. However, if we consider only stuck-at tests then both transitions (rising and falling) are not needed and test sequence (x, ¬x) of length t=2 would be sufficient.

In conclusion, by applying the test sequence X (or ¬X), we cover all stuck-at and delay faults at the interconnects and also opens/shorts between adjacent interconnect lines. Furthermore, the respective fault models in the data paths of the crossbar and buffers of the switch are also fully covered. We admit that we do not have information yet how thoroughly the respective fault models in the control part, mux address signals and register control signals are handled by the proposed approach.

5.7 Overview of Test Configurations

Note, that the biggest challenge in testing the switches embedded into NoC lies in the fact that the network can be accessed only from its external boundaries. The Subsections below explain the method in further detail.

The approach presented in this thesis is based on applying three test configurations to cover the entire switching network. In our previous paper [49] we have shown that by applying them we will achieve near 100 % fault coverage for the crossbar switch and the I/O registers. The configurations are shown in Fig. 41 and they include (a) straight paths, (b) turning paths and (c) resource connections, respectively. A configuration is set up by adjusting the corresponding destination address fields of the transmitted packets to the last row (column) of the network matrix.

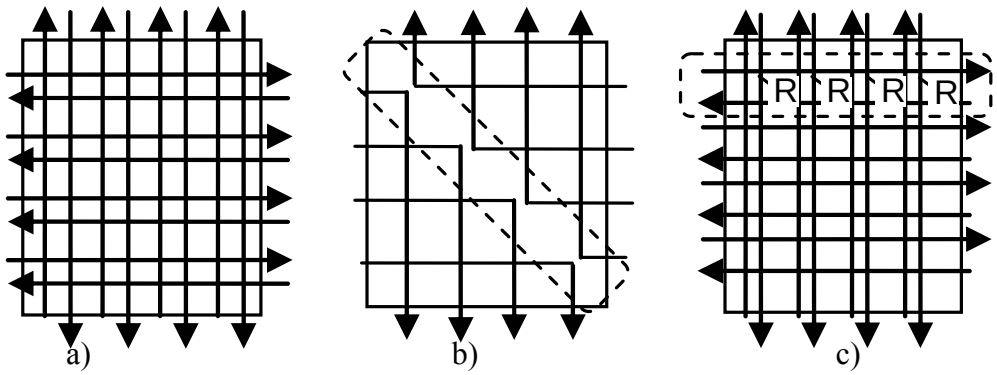


Figure 41: Test configurations for mesh-like NoCs

Configuration *a* will be set up by letting the packets pass straight through the network. This will cover the faults in the straight links of the network. A constraint is that each bit in the data bus must be traversed with a 0 and a 1 (i.e. toggle coverage must be 100%). Additionally, vertical and horizontally sent data must be distinguished from each other. This is necessary in order to cover faults in multiplexer addressing of the crossbar switch (See the previous subsection!). It takes d·n+1 test patterns to cover this configuration in meshes containing n×n switches, where d is the delay in the switch. For switches including both, input and output registers d is equal to 2. Note, that configuration *a* for bouncing switches takes 4(d·n+1) clock cycles because here, each direction has to be distinguished from the remaining three.

In configuration *b* we are taking advantage of the deterministic XY routing implemented in the switches under test. The packets will be sent by the X axis of the mesh and will meet at a diagonal of the switch. Here, the diagonal will be shifted over the entire network matrix until all the switches have been covered. The main issue behind testing the turning paths is that YX paths are normally not supported by the basic routing scheme and thus, a special test mode input and modification of switches control part has to be introduced in order to run configuration b for turns from Y to X axis. Configuration *c* is needed to cover the links to resources. In order to run this configuration a loopback in RNI has to be provided. In the following the three configurations are described in more detail.

Please note that in thesis we do not consider the issue of handling synchronizers, which are normally present in the NoC networks. A possible solution would be to repeat some of the test patterns in the test sequences multiple times to allow required test data sent from different directions be present at the switches under test simultaneously. More sophisticated solutions should take into account the exact layout of the synchronizers inside the network.

5.7.1 Testing straight paths: configuration a

As mentioned above, the testing of multiplexers in the switching network of NoCs is based on a functional fault model, where an adress value is selected and value at the input corresponding to it is distinguished from all the remaining mux input values. In order to test the multiplexer, tests for each address value has to be performed. Furthermore, due to the fact that switches are embedded into the NoC's network, dedicated test configurations have to be implemented similar e.g. to FPGA interconnect testing approaches [47]. However, differently from the latter in the external test approach presented in this thesis the test configurations are based on the routing algorithm implemented in the switches. For the current case it is the deterministic, or XY routing.

Let us at first take a look at the functional testing of straight paths in the NoC (configuration a). Fig. 42 explains that case. In the Figure, a network of n×n switches is presented. Arrows show transmission of data packets through the network. The following notation is used: black lines correspond to test pattern x (i.e. $\{01\}^{k/2}$) and grey lines correspond to pattern ¬x ($\{01\}^{k/2}$), respectively, where k is the width of the interconnect bus in the network.

In the case of non-bouncing switches it is sufficient to distinguish horizontal paths from the vertical ones (See Fig. 42.a). Here, test sequence X is transmitted from all the odd-numbered columns of the network and ¬X is sent from the even-numbered columns. For horizontal paths, we start transmitting ¬X via the odd-numbered rows and X via the even-numbered rows, respectively (Fig. 42.a).

The test configuration is organized by setting the destination address fields of the transmitted packets to the last row (column) of the network matrix. It takes n clock cycles for the test data to reach the last switch (n+1 for switches with input registers!) and additional t-1 clock cycles to apply the remaining patterns of the test sequence. Thus, n + 2 clock cycles are required in order to test all the straight paths in a NoC consisting of non-bouncing switches.

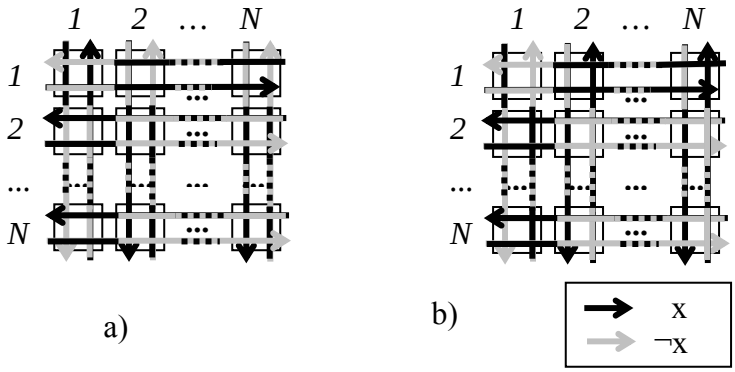


Fig. 42: Testing straight paths a) in non-bouncing switches, b) bouncing switches

In the case of bouncing switches a direction is distinguished from all other three. Fig. 42b shows the case where traffic from North to South is distinguished from the other input paths. There are four directions, thus, 4(n+t-1) clock cycles would be needed to perform full input value distinguishing for the network under test.

5.7.2 Testing direction changes in routing: configuration b

5.7.2.1 Deterministic routing (XY routing). The next configuration in the functional test is to cover changing points in the deterministic routing. Deterministic routing (or XY routing) is a strategy where packets are sent at first along the X axis of the network and then along the Y axis. In other words, only turns from X to Y axis are possible, not vice versa. Fig. 43 explains test configurations for such turning points in routing along the main diagonal of the network matrix. In order to cover all the routing direction changes in each diagonal has to be tested separately. The diagonal is shifted up with one configuration (W→N, E→S shown in Fig. 43a) and down with the remaining one (E→N, W→S in Fig. 43b).

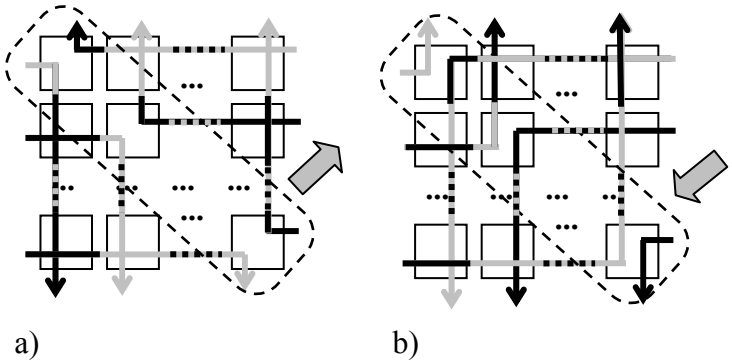


Figure 43: Testing direction changes in routing

Similar to testing straight paths explained in the previous subsection the configuration is set up by adjusting the destination address fields of the transmitted packets.

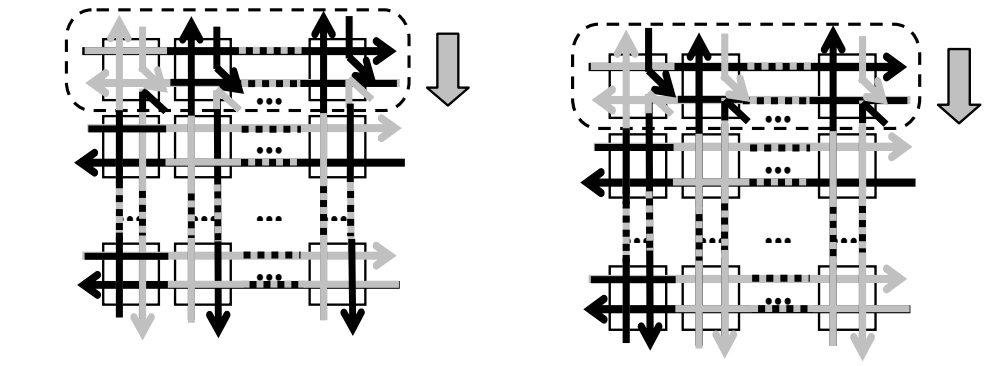
5.7.2.2 Inverse deterministic routing (YX routing). Note, that there is a problem as deterministic routing does not consider paths entering from Y axis and exiting to X axis. However, such routing scheme might be selected in case of conflicting packages inside the switch. Thus, in order to entirely cover the structural faults in the switch multiplexers, a dedicated test mode input for the switch can be added, which would temporarily enable inverse deterministic routing (YX instead of XY).

The test for the inverse deterministic routing is dual to the configuration explained in Subsection 5.7.2.1 (See also Fig. 43).

5.7.3 Testing resource connections

The next stage of the external test is sending test data to and from the resources and distinguishing it from data at the input directions N, W, E, S. This test configuration can be organized as follows (See Fig. 44). Let us assume that we start sending test data from N to R and from R to S. We pick rows with resources to be addressed. At first we distinguish data sent to resource from the other inputs. Then the data transmitted from the resource is distinguished. We show an example for sending packets from North to Resource and from Resource to South in Figure 44a and 44b respectively. In this example, packets whose test data values are different from the resource test data are sent from W and E. The row with targeted resources is then gradually shifted downwards.

The main concern with testing the resource interfaces is the question how a resource can initiate test sequences and how the test data sent to resources can be observed. This problem can be solved simply by implementing loopback of test data and readressing of test packets inside the Resource Network Interface (RNI) during the test mode. This will be explained in detail in Section 5.9.



a) b)

Figure 44: Testing traffic: a) coming from resources, b) entering resources

5.8 Improved Test Configurations

The test configurations described above provide for a very high fault coverage and require low silicon area overhead. However, configurations presented in Fig. 41b and 41c introduce a square time complexity. This is because of the fact that the main diagonal in these configurations hav to be shifted during the test. Therefore, as experiments shown in Section 6 indicate, test application becomes prohibitively slow for larger NoC networks. As an example, 100x100 mesh would require more than hundred thousand clock-cycles to run.

In this paper, we propose replacing the configuration discussed in section 5.7.2 by a different configuration, where test packets are forwarded along the mesh diagonals and forced to make turns at each switch. In addition, we improve configuration discussed in section 5.7.3 by allowing all the resource connections be tested in a single run. This adds a one cycle delay at each row of the mesh but avoids the square complexity incurred because of the need to shift the row of resource connections to be tested. As experiments show, this allows reducing the test length up to two orders of magnitude without any sacrifices to area overhead and fault coverage.

5.8.1 Testing direction changes in routing

The role of the test configuration *b* is to cover changing points in the deterministic routing. Deterministic routing (or XY routing) is a strategy, where packets are sent at first along the X axis of the network and then along the Y axis. In other words, only turns from X to Y axis are possible, not vice versa.

The limitation of the XY routing from the testing perspective is that it does not consider paths entering from Y axis and exiting to X axis. However, such routing scheme might be selected in case of conflicting packages inside the switch. Thus, in order to entirely cover the structural faults in the switch multiplexers, a dedicated test mode input for the switch can be added, which would temporarily enable inverse deterministic routing (YX instead of XY).
In [17], an approach was implemented, where routing changes were tested along the diagonals of the network matrix. In order to cover all the routing direction changes each diagonal has to be tested separately. The diagonal is shifted with the configuration (See Fig. 43a and Fig 43b). However, shifting of the diagonal causes the $O(n^2)$ complexity for test application time. It requires $8dn^2+16n$ clock-cycles to carry out configuration discussed in section 5.6.2.

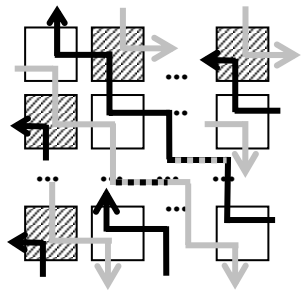


Figure 45: Testing direction changes in modified method

In this modification we propose a new configuration, which is similar to [17] relies on the possibility to switch the NoC routers into the inversed (i.e. YX) routing mode for testing. However, only half of the routers are switched toYX mode, while the others remain in normal mode. A checkerboard pattern is used here, as shown in Fig 45.
Only the striped switches in Fig 45 are in YX mode.

This does not require any additional test hardware with respect to [17]. However, the test time complexity will be only $8dn+8$, because of the fact that all the switches can be tested in a single run.

5.8.2 Testing resource connections

An additional stage of the external test is sending test data to and from the resources and distinguishing it from data at the input directions N, W, E, S. This test configuration can be organized as follows. Let us assume that we start sending test data from N to R and from R to S. We pick rows with resources to be addressed. At first we distinguish data sent to resource from the other inputs. Then the data transmitted from the resource is distinguished. We showed an example for sending packets from North to Resource and from Resource to South in Figure 41c. In this example, packets whose test data values are different from the resource test data are sent from W and E. In [11] an approach was used where the row with targeted resources (the first row of the network in Fig 41c) is gradually shifted downwards. In this paper, we improve the configuration by allowing all the resource connections be tested in a single run. This adds a one cycle delay at each row of the mesh but avoids the square complexity incurred because of the need to shift the row of resource connections to be tested. As a result, only $4(dn+1)$ clock-cycles are needed for the configuration as opposed to $4(d(n^2+n)+1)$ cycles in [17].
The main concern with testing the resource interfaces is the question how a resource can initiate test sequences and how the test data sent to resources can be observed. This problem can be solved simply by implementing loopback of test data and readdressing of test packets inside the Resource Network Interface (RNI) during the test mode. This will be explained in detail in Section 6.

5.9 DfT Structures for External Tests

In order to implement the test algorithm in a NoC network, various DfT structures were devised. These include resource loopback for testing the crossbar multiplexer of the resource connection, a modification to the control part to force YX routing, a compact logic BIST for the control unit and dedicated test logic for covering the enable signals of switch buffers. The DfT structures are briefly summarized below.

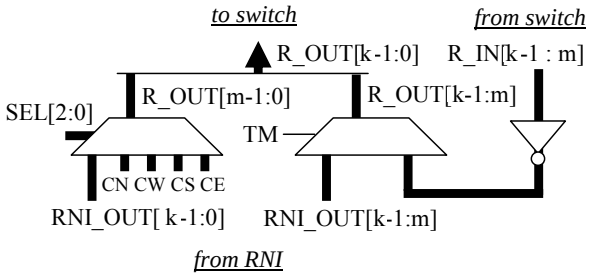


Figure 46: Loopback between RNI and the switch

5.9.1 Resource loopback

To carry out configuration *c* (See Fig. 41c), a loopback structure is needed for forwarding the test data back to the network in order to be observed. It consists of 2 multiplexers and an inverter array to allow distinguishing of the reflected test data. See Fig. 46 for the loopback of a k-bit total bus width including an m-bit address bus. The loopback is activated only in the test mode (TM signal high). In the functional mode the data is sent normally to and from the Resource Network Interface (RNI). Using a multiplexer we can choose from four address constants CN, CW, CS, CE, which are needed to submit packets to North, West, South and East directions, respectively. So all that is required for applying configuration c is to specify the value of select and switch the circuit to test mode. The total area overhead of the loopback is 0.9% of the switch.

5.9.2 DfT support for YX routing

In order to propagate the test patterns to cover all the paths in the NoC address decoder of the control path was modified to YX routing in the test mode. (XY routing may be covered in functional mode). The overheard area incurred is 0.4 % of the entire circuit.

5.9.3 Logic BIST for the control unit of the switch

The size of a 128-bit switch is 20 k NAND-gate equivalents. The control unit takes 10 % of the total switch area [6]. However, the test coverage of the controller is generally low in functional and/or pseudorandom testing (around 50 %). We have implemented logic BIST structures detecting 99.15 % of the single stuck-at faults in the control unit. The BIST was simulated using the freeware Turbo Tester system [50].
The area overhead compared to the size of the control unit is relatively large. The unit has 96 input/output pins. These pins are connected to flip-flops of registers, which can be reconfigured to linear feedback shift-registers. The resulting area overhead is around 500 gates, which makes as much as 25 % of the control unit area but only 2.5 % of the entire switch area.
Thus, the total area overhead required by the DfT structures is: 0.9 (loopback) + 0.4 (YX routing) + 2.5 (BIST) = 3.8 %.

5.10 Diagnostic Capabilities of the Test Configurations

5.10.1 Concept of switch links

Diagnosis is the process of locating the faults in a structural model of the Unit under test (UUT).This subsection discusses how we pinpoint the location of the faults in the switches.

In this thesis, we utilize a high-level functional approach to link faults. However, as experiments show, the fault model has very good correspondence with low-level structural faults. Let us introduce the concept of links and link faults in a NoC switch.
Definition 1. By *link* we understand a physical path between two I/O ports of the switch for sending packets through it.
Definition 2. If a packet enters a switch, but either faulty data or no data will be transferred to the addressed output port then we say the corresponding link is faulty, i.e. a *link fault* has occurred.

Note, that in the current approach we do not consider cases where the packet is not sent to the predicted output because of packet congestion. The configurations presented here are conflict-free. Furthermore, for the sake of simplicity we assume that only a single link in a system may be faulty at a time. Figure 47 presents the ten I/O ports of the switch and a link from North to East, shown by a dotted arrow, which we denote by N→E.

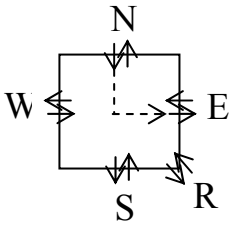


Figure 47: I/O ports of NoC switch and link N→E

There are five input and five output ports (directions). Thus there are 20 different links for the non-bouncing switches. (Note that bouncing switches are not considered in current diagnosis algorithm.) Let us assume that our NoC under test is a mesh-like network of n×m switches, i.e. we have column positions for the switch at the X-axis: $1 \le i \le n$ and row positions at the Y-axis $1 \le j \le n$. In the following we use a notation, where we denote by $E \rightarrow W_{i,j}$ a link fault at a connection from East to West inside a switch whose position at the X-axis is *i* and at the at the Y-axis is *j*.

5.10.2 Collapsing of link faults

In order to simplify the algorithm description but without any loss of generality the links to be considered in the diagnosis method are reduced to five equivalence classes shown in the table below

Table 3 Equivalence classes of NoC link faults

fault:	**equivalent faults:**	**reference:**	**conf.:**
E→W	W→E, S→N, N→S	vert.&horiz. links	*a*
E→S	W→N, E→N,W→S	XY turns	*b XY*
N→W	S→E, N→E, S→W	YX turns	*b YX*
N→R	E→R, S→R, W→R	resource inputs	*res. in*
R→S	R→W, R→N, R→E	resource outputs	*res. out*

In the leftmost column we present the link faults that are used as *representative faults* in the diagnosis method described below. The second column shows the faults equivalent to the chosen representatives. The third column gives the names of the link types and the final column shows the test configuration for targeting the link fault (See Fig. 41).

In the first three classes of link fault types the corresponding equivalent faults are dual to representative faults in a sense that they are detected by the same test configurations and can be regarded as mirrored cases of the tests for their representatives. In the two last rows we consider links to and from resources that can be managed by four configurations which are equivalent but differ in direction. This reduction in the fault classes represents the fault collapsing for link faults.

In the following we will consider locating the representative faults by utilizing the test configurations given in Fig. 41.

5.10.3 Diagnosis of link faults

The diagnosis method is explained basing on the diagnostic tree presented in Fig. 48. The diagnosis process proceeds as follows. First, we apply test configuration *a* for horizontal and vertical links in the network. The representative fault for these links is $E \rightarrow W$. If the test fails then we know that there is a fault at the row address *j*, but it is not clear what is the failing column *i.* However, if we apply configuration *b* for XY routing then by moving the main diagonal of the routing turns we can accurately point out the location of the faulty horizontal link. There are two possible outcomes. First, if the test fails until the diagonal shifts to a turning column position *k* then the fault is located at $E \rightarrow W_{k-1,j}$, otherwise, if configuration *b* passes then the fault is locatd at the last last column position, i.e. $E \rightarrow W_{n,j}$.

Let us now consider the case when configuration *a* passes. If then configuration b_{XY} fails then we can easily locate the exact position of the XY turn fault, whose representative is $E \rightarrow N_{i,j}$. This is because configuration *b* consists of XY turns and horizontal/vertical links but the latter class has been excluded from consideration by passed configuration *a*. If configuration b_{XY} passes then as a next step we diagnose faults in YX turns, which is handled similarly to XY turns.

Resource input links and output links are diagnosed by two separate configurations. Furthermore, four different directions per configuration must be handled because of the four input/output directions respectively. Nevertheless, this guarantees exact location of the respective link faults based on exactly similar properties as in testing the XY turns. In other words, the resource connections consist of horizontal/vertical links and resource input/output links. The first class is assumed to be fault-free at this point. By moving the row (column) with resource connections we are able to pinpoint the failing switch and link.

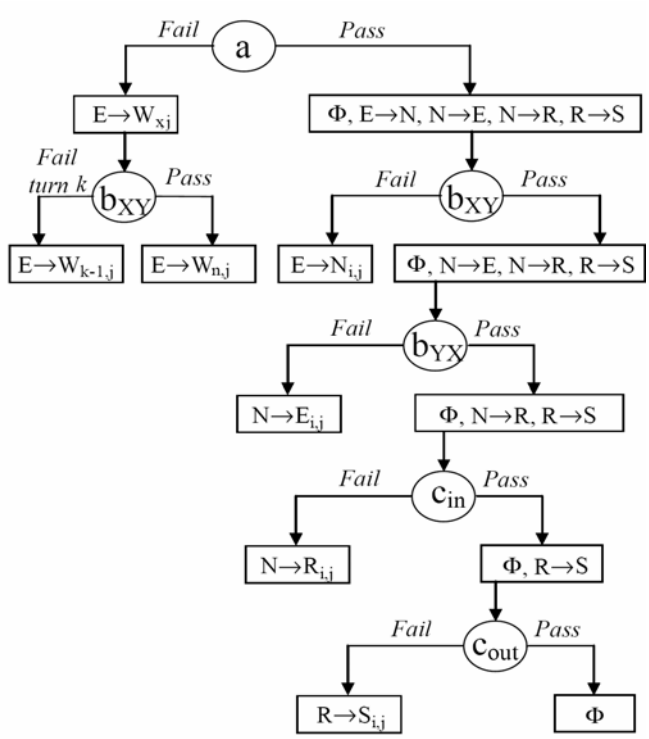


Figure 48: Diagnostic tree for n×m network

5.11 Summary

In this chapter we describe the orignal external test application method and a modified version which achieves a lower test application time as compared to the former along with the same high fault coverage.The next section covers the experimental results and conclusions.

Chapter 6
Experimental Results and Conclusions

About this Chapter

The experimental results and conclusions are discussed in this chapter. The first subsection elaborates experimental results in relation switch implementation results; this is followed by the fault coverage achieved by the external test method on 8 variants of the switch. The subsequent subsection deals with the modified version of test configurations which achieves a shorter test application time The next 2 subsections illustrate the fault coverage of the cross bars and the impact of DfT structures on the fault coverage. The final section elaborates the conclusion from the work.

6.1 Experimental Results

6.1.1 Switch Implementation Results

In order to evaluate the approach developed in this thesis a generic synthesizable VHDL description of a switch was created. In this description it is possible to select the width of the busses connecting the switches k and the type of the switch. For bus width k=128 was selected relying on the example of NOSTRUM switches [44]. The following implementation options could be selected: bouncing and non-bouncing switches, and-or and one-hot multiplexers and buffering of input data to registers. Total eight types of switches were synthesized. The area requirements of different components of the synthesized switch are shown in table 4.

Table 4: Area requirements of the switch

*-Control path components

Circuit	**Total Area**
*5 Ready_out DFFs	45
*5 Write_out DFFs	45
*5 Address decoders	420
*Priority sorter	213
Contr. signal decoder	150
Switch datapath	19955
Total area:	20928

6.1.2 Fault Coverage of Switches

The fault coverage was measured by a fault simulator belonging to the Turbo Tester system [50]. The results of the experiments are presented in Table 5. The first three colums of the Table specify the configuration of the switch (i.e. bouncing/non-bouncing, including/excluding input registers, and-or/one-hot encoded multipexers). The fourth and the fifth columns present the *single stuck-at* fault coverage numbers for the test configurations excluding and including test for inverse deterministic routing (YX routing) respectively. As it can be seen, forcing YX routing during configuration b increases the fault coverage by 7-9 per cent.
Experiments on these eight examples show that the fault coverage is higher than what can be achieved by a recently published DfT-based approach [41]. Furthermore, the test volume of the current approach is orders of magnitude lower than shown in [41]. An additional benefit of the method is that it supports at-speed test of delays and defects in the NoC network. The fault coverage of eight different versions of the switch is show in table 5.

Table 5: Experimental results on a family of 8 switches

Switch type			**Functional test configurations w/o YX routing**	**Functional test configurations with YX routing**
bounce	**input reg.**	**one-hot**	**fault coverage, %**	**fault coverage, %**
N	N	N	90.63	99.83
N	Y	N	92.38	**99.86**
Y	N	N	91.79	99.78
Y	Y	N	93.15	99.81
N	N	Y	90.10	99.26
N	Y	Y	91.90	99.37
Y	N	Y	91.18	99.09
Y	Y	Y	92.67	99.32

6.1.3 Impact of fault coverage on different crossbar implementations

In addition, the impact of crossbar multiplexer implementation was analyzed by experiments. Table 6 presents the analysis of stuck-at fault coverage measured in different types of switch crossbar implementations. Column 1 lists the switch type, which can be bouncing (“feedback”) or non-bouncing (“no fb.”). The alternative muxes used in the crossbar include binary encoded (“binary”) and one-hot encoded (“1-hot”) types. For the latter, two versions were synthesized: “structural” and **“**behavioral”. “Structural**”** was described as a two level logic gate implementation, while “behavioral” was synthesized from the “case” statement adding a small decoder to the logic.

Table 6: Results on different crossbar implementations

Crossbar implementation	# of faults	# of tested	Fault coverage, %
binary, no feedback	**11640**	**11640**	100.00
binary, feedback	**14240**	**14106**	**99.06**
1-hot, no fb. behav.	**11790**	**11685**	**99.15**
1-hot, fb. Behavior.	**14460**	**14106**	**98.21**
1-hot, no fb, struct.	**11560**	**11560**	100.00
1-hot, fb. structural	**14130**	**12205**	**86.37**

As it can be seen, the test approach allows 100 % stuck-at fault coverage for non-bouncing switches, whose crossbar has been implemented with binary encoded muxes. For one-hot encoded implementation the logic level solution guarantees full coverage. Although, the proposed configurations did not explicitly target the feedback links of the bouncing switches we did measure the coverage for them. As a result, 99.06 % coverage was achieved for binary encoding and a coverage 98.21 % was achieved for one-hot (behavioral) implementation of feedback switches.

6.1.4 Impact of modified configurations on test application time

The results of the test application time measurements are presented in Table 7. Note, that the table shows four, not eight types of switches because the implementation details of the muxes do not affect the test time. As we can see, there is not much difference between the two methods for smaller networks. However, for 100x100 meshes the test application time is two orders of magnitude shorter as in [5,49].

Table 7: Test application time in clock cycles

Switch type	3x3 network		10x10 network		100x100 network	
	[11]	new	[11]	new	[11]	new
Non-bounce/ Not buffered	176	**52**	1415	**143**	122105	**1313**
Non-bounce/ Input buffer	299	**91**	2665	**273**	242605	**2613**
Bouncing/ Not buffered	188	**64**	1448	**176**	122408	**1616**
Bouncing/ Input buffer	320	**112**	2728	**336**	243208	**3216**

6.1.5 Effect of DfT structures

A network of 3x3 switches has been implemented in order to evaluate the effect of DfT structures proposed. Table 8 presents the fault coverage results and circuit area in the 128-bit and 512-bit versions of the NoC network. Area is shown for a single switch in the 3x3 network and measured in nand-gate equivalents. The coverage values represent single stuck-at fault coverage as measured by Turbo Tester sequential fault simulator [50]. This information includes the DfT structures, except for rows denoted by the astrix (*) character, which do not include the logic BIST of the control part. Rows 'Reg_in', 'Reg_out' and 'Mux' correspond to input buffers, output buffers and crossbar multiplexers of the switch datapath, respectively. Rows 'Sorter', 'Address decoder' and 'Output decoder' show the fault coverage and area requirements of the control unit blocks: priority sorter, address decoder and control signal decoder, respectively. Rows 'CP no BIST' and 'CP w BIST' show the fault coverage and area of the control part without and with logic BIST structures, respectively.

Table 9 shows the impact of implementing the DfT structures for external testing. The results can be inferred as following. The usage of resource feedback structure improved the fault coverage by 3 and 5 % for the 128 and 512 bit networks, respectively. The control part was tested using a BIST simulated by the freely available Turbo Tester software [50]. This improved the coverage by further 5 % and 2 %, respectively. The overall stuck-at fault coverage obtained for the 128 bit version was 97.54 % and that of the 512 bit version was 99.33 %.

Table 8: Experimental results without control part BIST

*Coverage before application of BIST

Block	F.C. (128), %	F.C. (512), %	Area (128), gates	Area (512), gates
Reg_In	97.3	99.3	6050	25605
Reg_Out	97.9	99.9	6050	25605
Mux	97.1	99.2	3230	12830
Sorter*	32.8	34.7	635	635
Address decoder*	50.7	50.7	565	565
Output decoder*	64.1	53.6	1075	1075
CP, no BIST*	48.32	50.22	2175	2175
CP, w BIST	99.14	99.14	2700	2700

Table 9: Impact of the DfT structures

DfT structure	Fault coverage (128), %	Fault coverage (512), %
No DfT	89.70	92.68
Resource loopback	92.19	97.93
Resource lb. + BIST	97.54	99.33

As it was mentioned above, the proposed approach has linear test time requirements with respect to the size of the NoC network. Note, that when relying on the test configurations we are testing multiple switches in parallel. Furthermore, we are not concerned about test access and excessive amount of test data as in the case of traditional scan-based approaches. Our method consumes 320 clock-cycles for the 3x3 network, which are about two orders of magnitude less than the scan-based approaches from the literature [41]. Similar to the scan approach, the test length grows linearly with the size of the circuit.

6.2 Conclusions

The work defines a scalable functional test method for NoC switching networks. The proposed algorithm allows to cover nearly all of the single stuck-at faults in the switching networks, transition faults, opens and shorts at the interconnect lines. A generic parametrizable VHDL description of a deflecting NoC switch was implemented and a benchmark family of 8 switches representing different possible architecture configurations was synthesized.

The test patterns are applied externally from the boundaries of the switch as opposed to using scan paths and wrappers for test access. A 3*3 network was synthesized and the data width of the Switches in the Network could be either 128 or 512.Various test configurations were used to cover the entire switching network. The initial method produces a test set whose volume has a square complexity with respect to the rank of the network matrix.

The thesis also defines a functional fault model for testing crossbars in deflecting switches. The fault model was organized into test configurations that allowed targeting of switches embedded into regular mesh-like networks. The impact of crossbar multiplexor on fault coverage was analyzed. The highest coverage was 100%, while the lowest being 86, the average overall coverage being 97.1 %.

A new high-level concept of faulty links (directions) in NoC switches was introduced. One of the novelties of the thesis is to propose the use of the functional test configurations with a goal to locate faults in individual links of the switches. Collapsing of link faults based on equivalent configurations was also proposed. As a

result, a method was proposed that is capable of unambiguously diagnose a link fault in the network in a very short test application time. Experiments showed that, although working at higher abstraction levels, the method has very high coverage for logic-level structural faults. The test configurations were modified to get a much shorter test time. Test time requirements are square root of the number of switches and the supporting testability structures occupy only up to 4 per cent of overhead silicon area. Experiments on different NoC setups showed that the new method is well applicable for large networks.

In addition, the work proposed a set of Design-for-Testability (DfT) techniques for application of test patterns from the external boundary of a Network-on-a-Chip (NoC). In previous papers we have proven that such external test configurations provide for a high-fault coverage, diagnostic resolution and scalability in testing mesh-like NoCs. We have implemented a parametrizable switching network and developed a set of DfT structures to support testing of network switches using external test configurations. The proposed DfT structures include resource loopback for testing the crossbar multiplexer of the resource connection, a modification to the control part to force YX routing and a compact logic BIST for the control unit.

The main novel contribution of the approach presented in the thesis is to combine the test configurations for NoC developed by the authors in [6] and the new method for locating faults in the NoC interconnection infrastructure [5] by area efficient DfT structures promoting the quality of test for NoC networks. Experiments show that the proposed structures allow near-100-percent test coverage at the expense of less than 4 % of extra switch area.

References

1. The International Technology Roadmap for Semiconductors, http://www.itrs.net.

2. Networks on Chip by Hannu Tenhunen and Axel Jantsch, Kluwer Academic Publishers

3. A. Alaghi, N. Karimi, M. Sedghi, and Z. Navabi, Online NoC switch fault Detection and diagnosis using a high level fault model", in Proceedings of the 22nd IEEE International Symposium on Defect and Fault- Tolerance in VLSI Systems (DFT), 2007.

4. Aktouf, C. "A Complete Strategy for Testing an on-chip Multiprocessor Architecture". IEEE Design & Test of Computers, vol. 19-1, 2002, pp. 18-28

5. Jaan Raik, Vineeth Govind, Raimund Ubar. DfT-based External Test and Diagnosis of Mesh-like NoCs. IET Computers and Digital Techniques, 2009.

6. J.Raik, V.Govind, R.Ubar. An External Test Approach for Network-on-a-Chip Switches. Proc. of the IEEE Asian Test Symposium 2006, pp. 437-442, Fukuoka, Japan, Nov. 2006

7. Kim Petersén, Johnny, Bengt Magnhagen. Towards an Almost C-Testable NoC Test Strategy. IEEE East-West Design and Test Symposium, 2007.

8. M. Renovell, J.M. Portal, J. Figueras, Y. Zorian, "Testing the Interconnect of RAM-based FPGAs", IEEE Design & Test, January-March 1998, pp. 45-50.

9. http://www.design-reuse.com/articles/10496/a-comparison-of-network-on-chip-and-busses.html

10.Open Core Protocol International Partnership,http://www.ocpip.org

11.www.arm.com/products/solutions/axi_spec.html

12. Interconnect-Centric Design for Advanced SOC and NOC, Nurmi.J, Tenhunen H., Isoaho J., Jantsch A.

13. Interconnection Networks: An Engineering Approach by Jose Duato (Author), Sudhakar Yalamanchili (Author), Lionel M. Ni

14. Networks on Chips: Technology and Tools by Giovanni De Micheli

15. Networks-on–chips: Theory and Practice, Fayez Gebali, Haymtham Elmiligi

16. Low-Power NoC for High-Performance SoC Design by Hoi-Jun Yoo, Kangmin Lee ,Jun Kyoung Kim

17 http://techresearch.intel.com/articles/Tera-Scale/1449.htm

18 http://www.iis.ee.ethz.ch/async/

19 http://ralyx.inria.fr/2007/Raweb/vasy/uid112.html

20. VLSI Test Principles and Architectures: Design for Testability (Systems on Silicon) by Laung-Terng Wang, Cheng-Wen Wu, and Xiaoqing Wen

21. Advances in Electronic Testing: Challenges and Methodologies (Frontiers in Electronic Testing) by Dimitris Gizopoulos

22. Design-For-Test for Digital IC's and Embedded Core Systems by Alfred Crouch

23. J.P.Roth "Diagnosis of Automata Failure; A Calculus and a method", IBM Journal, PP 278-291 (1966-7)

24. P .Goel An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits, IEEE Transactions on Computers Volume 30 , Issue 3 (March 1981)

25. H. Fujiwara, T. Shimono, On the acceleration of test generation algorithms, IEEE Transactions on Computer(1983)

26. M.Schulz, E.Trischler and T.Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," IEEE Trans. on CAD, vol.7, No.1, Jan. 1988, pp.126-137.

27. VLSI Testing :Digital and mixed analogue/digital techniques,Stanley .L.Hurst

28. Testing of digital systems by Niraj K. Jha, Sandeep Gupta,Cambridge university press

29. Handbook of Testing Electronic Systems by O.Novak, E.Gramatova, R.Ubar.

30. Arithmetic Built-in Self –Test For embedded Systems by Janusz Rajski,Jerzy Tyszer

31. Digital Systems Testing & Testable Design by Miron Abramovici, Melvin A. Breuer , Arthur D. Friedman

32. E. J. Marinissen, A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores (1998) , Proc. Int. Test Conf

33. Vermeulen B., Dielissen J., Goossens K., Ciordas C., Bringing communication networks on a chip: test and verification implications; Communications Magazine, IEEE, Volume: 41 , Issue: 9 , Sept. 2003

34. Dey, S., Xiaoliang Bai , Yi Zhao, Fault modeling and simulation for crosstalk in system-on-chip interconnects, Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design

35. Grecu.C, Ivanov.A., Saleh.R,Pande, Testing Network-on-Chip Communication Fabrics, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on Computers, Dec 2007

36. Andrei Rădulescu, John Dielissen, Kees Goossens, An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration," date, vol. 2, pp.20878, Design, Automation and Test in Europe Conference and Exhibition Volume II (DATE'04), 2004

37. Pande.P, Grecu.C, Ivanov.A, Saleh. R.A , Switch-based interconnect architecture for future systems on chip, he 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications (IWSOC'03), 2003

38. C. Liu, E. Cota, H. Sharif and D. K. Pradhan, Test Scheduling for network-on-chip with BIST and precedence constraints, in Proc. Int. Test Conf, October 2004.

39. C. Aktouf, A complete strategy for testing an on-chip multiprocessor architecture, IEEE Design & Test of Computers, 19(1), January/February 2002.

40. L. Chen, S. Ravi, A. Raghunathan, and S. Dey, Scalable software-based self test methodology for programmable processors, in Proc. ACM/IEEE Design Automation Conf., June 2003

41. A. M. Amory, E. Briao, E. Cota, M. Lubaszewski, and F. G. Moraes, A scalable test strategy for network-on-chip routers, in Proc. Int. Test Conf., November 2005

42. System-on-Chip Test Architectures: Nanometer Design for Testability by Laung-Terng Wang , Charles E. Stroud , Nur A. Touba

43. S. Kumar, et al., A Network-on-Chip architecture and design methodology. Proc. IEEE Comp. Soc., Apr. 2002

44. Millberg, M.; et al.. Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip, Proc. DATE, Vol.2, 2004.

45. J. Raik, R. Ubar, "Fast Test Pattern Generation for Sequential Circuits Using Decision Diagram Representations." Journal of Electronic Testing: Theory and Applications, Kluwer, Vol. 16, No. 3, , June, 2000

46. Makar, S.R., and E.J. McCluskey, "On The Testing Of Multiplexers," Proc. 1988 Int. Test Conf., Washington, DC, September 12-14, 1988.

47. M. Renovell, J.M. Portal, J. Figueras, Y. Zorian, "Testing the Interconnect of RAM-based FPGAs", IEEE Design & Test, January-March 1998,

48. P.T. Wagner, "Interconnect Testing with Boundary Scan," Proc. Int'l Test Conf. 1987, IEEE Press, 1987, pp. 52-57.

49. J.Raik, V.Govind, R.Ubar. An External Test Approach for Network-on-a-Chip Switches. Proc. of the IEEE Asian Test Symposium 2006. Fukuoka, Japan, Nov. 2006

50. http://www.pld.ttu.ee/tt/

51. K. Kariniemi, J. Nurmi, Fault tolerant XGFT network on chip for multi processor system on chip circuits. *Proc. of FPL*, pp. 203 – 210, 24-26 Aug. 2005.

Curriculum Vitae

Personal data

Name	Vineeth Govind
Date and place of birth	20.10.1979, Kerala, India
Citizenship	Indian

Contact data

Address	Raja 15, Tallinn 12618, Estonia
Phone	+372 620 2259
E-mail	Vineeth@pld.ttu.ee

Education

2005-……	Ph.D. student in Computer Engineering, Tallinn University of Technology
2002-2004	M.Sc. in Electrical Engineering with Specialization in System on Chip Design, Royal Institute of Technology Stockholm, Sweden
1997-2004	Bachelors in Electronics and Communication Engineering, M.S University, Tamil Nadu India

Awards

2007	Scholarship of Estonian Information technology Foundation (EITSA)
2006-2009	Scholarship of Archimedes foundation

Curriculum Vitae

Isikuandmed

Nimi	Vineeth Govind
Sünniaeg ja -koht	20.10.1979, Kerala, India
Kodakondsus	India

Kontaktandmed

Aadress	Raja 15, Tallinn 12618, Eesti
Telefon	+372 620 2259
E-post	Vineeth@pld.ttu.ee

Hariduskäik

2005-……	doktorant, Arvutitehnika instituut,Tallinna Tehnikaulikool,
2002-2004	Magistrikraad elektroonikas kiipsüsteemide erialal Stockholmi Kuninglik Tehnikakõrgkool (KTH), Stockholm, Rootsi
1997-2004	Bakalaureusekraad elektroonikas ja tele-kommunikatsioonis, M.S. University, Tamil Nadu, India

Teaduspreemiad

2007	Eesti Infotehnoloogia Sihtasutuse (EITSA) stipendium
2006-2009	SA Archimedes stipendium